TOC

About This Guide

This is an archived copy of the v23 documentation provided for Logi Info v23.3 and its service packs.

Notice: Archived Documentation

This documentation is provided as a courtesy reference for a version of our software that is no longer under active development or support. The information contained herein is offered without warranties of any kind, either expressed or implied, including but not limited to warranties of accuracy, completeness, or fitness for a particular purpose.

While this archived material may assist with understanding historical functionality, please be aware that the software described is no longer maintained at this version level and may contain outdated or inaccurate information. Images may not reflect currently supported modules, support sites, or third party products. This software may not be compatible with current versions of previously compatible third party products.

To access and upgrade to current software solutions and receive ongoing support, please contact our customer support team. They can assist you in migrating to the latest appropriate software version that meets your needs. Our support team is available to help ensure a smooth transition to actively maintained alternatives that provide the functionality and reliability you require.

Logi Info v23.1 Overview

Logi reports deliver data in a **presentable** and **accessible** manner as web pages. They use Internet technologies and a browser to distribute rich, interactive reports and applications. Logi Info brings leading-edge Internet technologies together, allowing you to quickly deliver results efficiently.

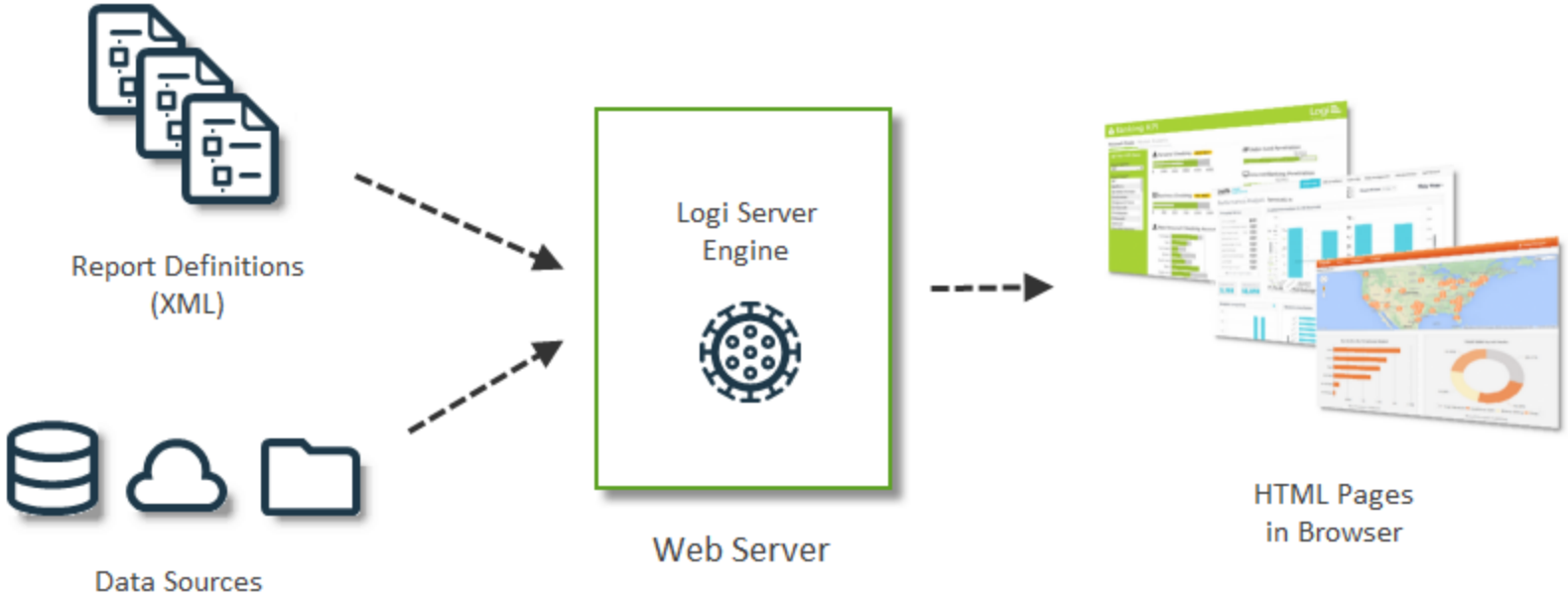
The Logi Info v23.1 Help System provides you with everything you need to know to fully utilize all of the features of Info v23.1. This topic provides a brief explanation of Info v23.1, and you can access other relevant information using the links we have listed in this topic, by looking at the Table of Contents, or by using Search.

This introductory topic is broken into the following topics:

If you are already familiar with Info and want to read up on the latest updates, see: Logi Info v23.1 Release Notes and Logi Info v23.1 Enhancements.

For information about hardware and software requirements for Logi Info v23.1, see: System Requirements - Logi Info.

Logi Info v23.1 provides a robust feature set to meet your Analytic needs.



A Logi *application* consists of one or more *reports*. A "report" is really a web page so, while Logi applications are most frequently used as reporting and analysis tools, they can also be used to create full-featured websites. The framework underlying this consists of:

- **Logi Studio**, an integrated development environment.
- **Report Definition** text files, which contain source code (in our XML-based coding language).
- The **Logi Server Engine**, a web server extension.

If you're a developer who's just getting started with Logi Info, you may not be aware of all of the features available in the products. If you'd like a visual introduction to the elements, you can go back into DevNet to see our Element Gallery.

DevNet also includes a number of free sample applications that you can download and run in Studio to see the techniques used to produce a variety of reporting applications.

When you are ready to get started with Logi Info, see "Get Started with Logi Info v23.1" on page 35.

Here are categorized links to relevant topics for many features:

Category	Description
Logi Studio	Studio is our development tool. Your next step should be learning about Studio's features (see "Introducing Logi Studio" on page 49) and how to use them (see *Using Logi Studio*).
Managing Your Product	Updates for Logi Info are available monthly. Keep abreast of the *Current Releases* and the Release Notes so you'll be up-to-date.
Designing Reports	Logi Info helps you understand how to design a basic page and use elements to achieve the desired result.
Accessing and Retrieving Data	The first step in reporting is often getting the data. Logi Info is able to get data from a large number of data sources. First you need to connect to a data source (see *Datasource Connections*) and then you need to retrieve its data (see *Datalayer Introduction*).
Tabular Data	Some data is best reported in tables and we have a variety available, including the regular Data Table (see *Data Tables*), the crosstab (or "pivot") table (see *Crosstab Tables*), *Multi-Column Lists*, and data lists (see *Data Lists*).

Category	Description
Charts	Charts are a great way to present data, and we have traditional static and animated charts (see *Chart Canvas Charts*). We also have non-traditional charts such as *Heat Maps*, polar charts (see *Polar Charts*), and text clouds (see *Work with Text Clouds*).
Dashboards	Dashboards, with the multiple panels, provide a great way to see a variety of data in a glance, and our Dashboards offer runtime flexibility and customization, if desired. For more information, see *Logi Info Dashboard* .
Tabbed Panels	A excellent method of organizing information on the page is to use *Tabbed Panels*.
Maps	Associating data with maps is a popular technique and we offer animated maps. For more information, see *Animated Maps*, *Google Maps* and, in v12.5+, *Leaflet Maps*.
Super-Elements	Logi Info includes a set of "super-elements", which are self-contained elements that produce tables and charts, and that have their own user-interface, so users can manipulate them at runtime. They include *The Analysis Chart*, *The Analysis Filter*, Analysis Grid (see *The Analysis Grid for End Users*), Chart Grid (see *The Chart Grid*), Dimension Grid (see *Dimension Grid Wizard*), OLAP Grid (see), and *Report Author*.
Presentation	The appearance of web pages is governed by style classes. We offer built-in Themes and the ability to work with your custom style sheets (see *Style Sheets*). In addition, you can add HTML tables (see *HTML Tables*) and divisions (see *Divisions*) and *Format Data* to get the desired look. We also support *Doctype Declarations*.
Showing & Hiding	Our dynamic reporting approach allows your to show and hide sections of reports based on data, security roles, or other criteria. Show Modes (see *Show Modes*) and Conditions (see *Work with Conditions*) can be used to control element visibility.
Passing Information	Information and data can be passed from one report definition or (in Logi Info) process task to another. For more information, see *Pass Information*.

Category	Description
User Input	We provide a range of HTML-based user input elements and validation, so that users can control reporting dynamically at runtime. For more information, see *User Input Elements*.
Pop-up Menus and Pop-up Panels	Users can also interact with a variety of pop-up menus (see *Work with Popup Menus*) and *Shared Elements* at runtime.
Shared, Reusable Code	Some code, such as headers and footers, might appear in multiple reports, and it can be created as *Shared Elements*, which are easily reused and maintained in a single place.
Exporting Data and Reports	We make it easy to export your data and reports to a variety of formats, including Word (see *Export to Native Word*), *Excel Template* , CSV (see *Exporting to CSV File*), PDF (see *Export to Adobe PDF*), and XML (see *Exporting to XML*).
Printing Reports	It's also possible to, adding pagination, special headers and footers, and other features.
Templates	You may need to fill forms using data, see *Form-based Reporting*. We offer features that make it easy to create report definitions that will fill-in Word (see *Export to Native Word*), *Excel Template* , and PDF templates (see *Export to Adobe PDF*).
Security	Logi security offers a variety of levels of security granularity and lets you work with network security such as Windows Domains and LDAP. For more information, see *Intro to Logi Security* .
Scheduling	The Logi Scheduler allows report generation and distribution to occur at regular intervals. For more information, see *Logi Scheduler*.

Category	Description
Process Tasks	You can also make use of *Process Tasks* to provide automation and conditional processing.
Web Services	Logi reports are capable of interacting with web services, including Salesforce (see *Work with Salesforce*), Twitter (see *Work with Twitter*), and those you may write.

About *Elemental Development*

Elemental Development is the process of creating flexible reporting applications using predefined, XML-based objects or "elements".

The advantages of Elemental Development are:

- XML **elements** are reusable and encapsulate specific functionality common to most web applications.
- A hierarchical layout of elements makes it easy to manage the presentation and functionality of large web-based reports.
- Logi Studio's toolbox of elements eliminates repetitive coding, shortens development time, and minimizes errors.
- Elements can be easily added, deleted, and moved to create almost any type of report. Their **attributes** are easily configurable.

General Development Process

The general development process for a simple Logi application is:

1. Use Logi Studio's New Application wizard to create a new Logi application and register it with the web server.
2. In the new application's "The _Settings Definition" on page 78 (created by default), add a Connection to the desired data source.
3. In the Default Report definition, add elements to design the web page, and retrieve and manipulate the data.
4. Add elements to analyze and visualize (tables, charts, maps) the retrieved data.
5. Preview your work right in Logi Studio as you proceed and use its debugging features to troubleshoot problems.
6. If desired, extend the application with exports, security, user interface controls, and more.

Logi Studio includes a variety of wizards which make these tasks easy.

Data Retrieval Technologies

The traditional Logi Info method of data retrieval generally uses a **Connection** element to connect to a datasource, and a related **DataLayer** element to issue a query or request for data. The returned data is cached in temporary files and/or memory, and can be modified and extended (grouped, filtered, etc.) in place using other elements, before being made available for analyses and visualizations. Some datalayers for file system-based data do not need to use a Connection element.

Standard Logi Application Folders

A Logi application is comprised of Logi Server Engine files, *definition* files, and *support* files, stored in specific folders, under an application root folder. Development work consists primarily of creating definition files and adding support files.

 In keeping with standard web server file security conventions, files in folders under the Logi application root folder are "available" to the application, meaning that the account the web server uses to run the application has file access permissions in these folders. Files *outside* the root folder are generally not available.

You *must not* modify the names or contents of these standard folders:

- **bin** - The Logi Server Engine files for a .NET application.
- **assemblies** - Files that map .NET modules into Java servlets (Java application only).
- **rdTemplate** - Component definitions, style, and scripting files used by the Logi Server Engine.
- **rdDataCache** - Temporary data cache files (this folder is created automatically when the application runs).
- **rdDownload** - Temporary files created by the application (also created automatically).
- **WEB-INF** - The Logi Server Engine files for a Java application.

You may notice that each Logi Info application contains its own set of Logi Engine files and this makes the size of each application fairly large. Why not provide one copy of the engine files in a central location? When each application has its own engine files, there are no problems distributing or running it when *multiple applications of differing versions* are run on the same web server.

You *can* create and modify files in these standard folders, but *do not modify* the actual folder names:

- **_Data** - Appears when a special Logi definition is created that makes an application a JSON data provider.
- **_Definitions** - (Required) Report, process, and settings files that you create. Standard subfolders include _Processes and _Reports, which are part of every application.
- **_Metadata** - Appears when the Web Metadata Builder is used to create metadata files.
- **_MobileReports** - Appears when you create special mobile report definitions.
- **_SupportFiles** - (Required) Image, style sheet, HTML, JavaScript, and XML files you import or create.
- **_Templates** - Appears when you create special definitions for use with Word, Excel, and PDF templates.
- **_Themes** - Used when custom Themes are added to the application.
- **_Plugins** - Used when plug-in .DLLs and/or Java files are part of the application.
- **_Widgets** - Appears when you create special Logi Widget definitions.

You can also add custom folders beneath the application root folder, for example, to contain data or exported files. Take care when creating these folders to ensure that they're given the same file access permissions as the root folder (permissions aren't always inherited, especially when copying files and folders).

Standard Logi Application Files

You'll find these standard files in an application root folder; *do not* remove, rename, or alter them:

- **rdChart.aspx** - Builds and displays charts
- **rdPage.aspx** - Displays report definitions
- **rdProcess.aspx** - Processes tasks contained in process definitions
- **rdErrorLog** - Error logs (only present if Error Logging has been turned on)
- **rdDebug.aspx** - Generates and displays debug output (made obsolete in v10.0.259+).

Each Logi application also includes several standard files that you *can* customize, described below:

Default.aspx

The default web page registered with the web server is Default.aspx:

```
<%@ Page Language="vb" %>
<%
Dim sQueryText as String
If Request.RawUrl.indexof("?") <> -1 Then
sQueryText = Request.RawUrl.substring(Request.RawUrl.indexof("?"))
End If
Response.Redirect("rdPage.aspx" & sQueryText)
%>
```

The code above redirects users to rdPage.aspx; the main web page for displaying report definitions. You can replace the contents of Default.aspx with your own HTML. Any query string parameters submitted with the URL are also passed to rdPage.aspx. For example, the following links take viewers to the same location:

```
http://www.Logiapps.com/LogiReportSamples/WebService/Default.aspx?ZipCode=90210&Miles=25
```

```
http://www.Logiapps.com/LogiReportSamples/WebService/rdPage.aspx?ZipCode=90210&Miles=25
```

Global.asax:

You can create your own Global.asax files for integrated Logi applications.

```
Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
 Dim rdMaint As New rdMaint.Maintenance()
 rdMaint.AppStart()
End Sub

Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)
 Dim rdServer As New rdServer.rdSession()
 Call rdServer.SessionStart()
End Sub
```

The Application_Start and Session_Start event handlers are customizable but *must* be present and, at a minimum, contain the code shown above.

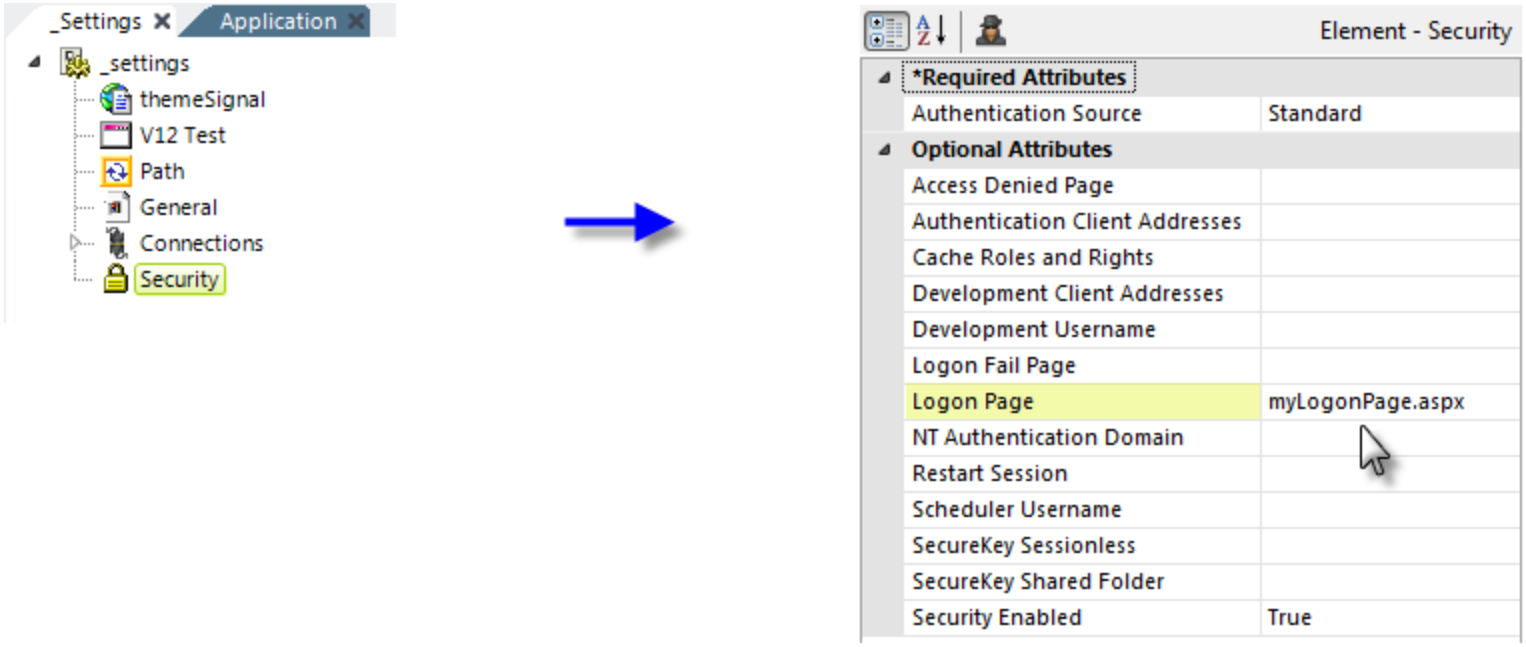
rdLogon.aspx:

All Logi applications are provided with a basic login page, which is used if Logi Security is enabled:

However, you can completely customize the rdLogon.aspx page and even rename the file. The following HTML tags for the login form are required to be present:

- `<form id=frmLogon action='<%=Session("rdLogonReturnUrl") %>' method=post>`
- `<input id="rdUsername" type="text" size="20" name="rdUsername">`
- `<input id="rdFormLogon" type="hidden" name="rdFormLogon" value="True">`
- `<input id="rdPassword" type="password" size="20" name="rdPassword">`
- `<%=Session("rdLogonFailMessage") %>`

Assuming that Logi Security is being used, you can present a custom logon page by configuring it in _Settings:



1. Open the _Settings definition to view its elements, as shown above.
2. Select the **Security** element.
3. Select the **Logon Page** attribute and enter the filename of your custom login page.

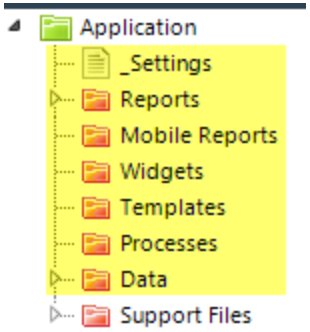
Web.config:

The Web.config file contains many useful settings for the entire Logi application. However, the Web.config file for a Logi application is *not the standard version* found in most ASP.NET applications and replacing it with a standard file will produce unpredictable results. Therefore, integrating a Logi application with another ASP.NET application has to be done in such a way that their Web.config files do not interact with or supercede each other. Generally speaking, we *do not* recommend that such an integration be accomplished by physically merging the application files and folders.

Having said that, the Web.config file is an XML file that you can modify to customize authentication, impersonation, globalization, session-state settings, error messages, and more. For Logi .NET applications, many configuration settings made in IIS Manager are written directly into Web.config, and any customized settings in Web.config always override similar settings in IIS Manager. More information about the Web.config file is available at this Microsoft web page.

Definition Files

In Logi Info, elements are combined in *definitions* to build web-based applications. One *report definition* is equivalent to one dynamic web page. Every Logi application contains at least one report definition and can include multiple definitions as the application grows.



The example above shows Logi Studio's Application Panel and the seven major types of definition files found in Logi applications, which are described below:

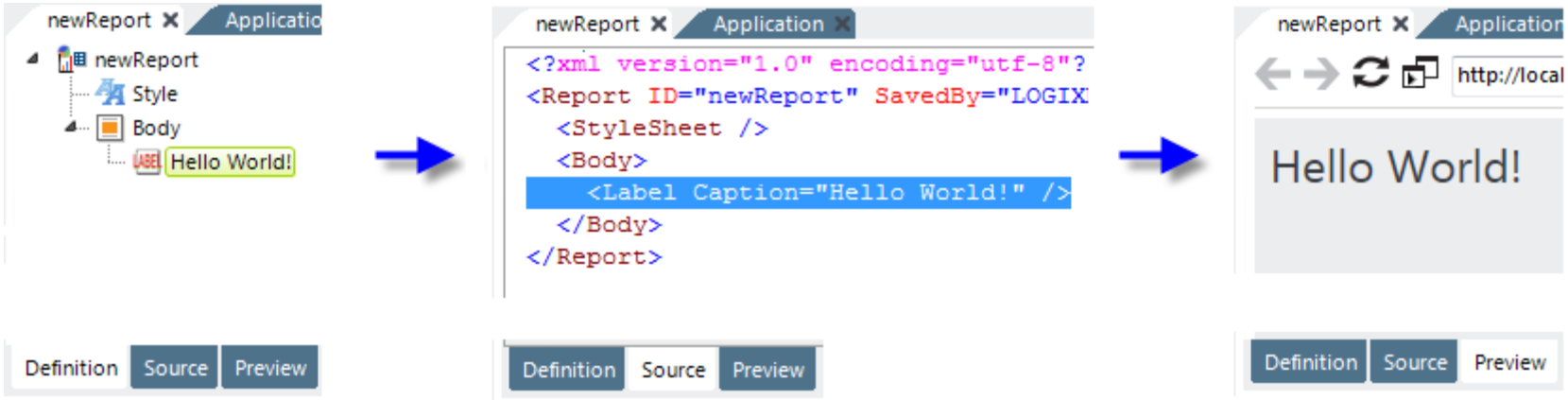
Folder/Definition	Description
_Settings	(Required) The one "The _Settings Definition" on page 78 in every Logi application contains global configuration values for the entire application. It contains elements that define the application's virtual path,

Folder/Definition	Description
	security information, and more.
Reports	Contains report definitions built with elements that define a dynamic web pages. A report definition typically contains report header and footer elements and a body element. The main body of the report can include any combination of text, charts, Dashboards, Data Tables, user input controls, etc. In the file system, this folder is `<appRoot>/_Definitions/_Reports`
Mobile Reports	Contains report definitions that are used exclusively to deliver content to mobile devices, built with a combination of common and special-purpose elements. In the file system, this folder is `<appRoot>/_Definitions/_MobileReports`
Widgets	Contains definitions for a special class of Logi reports that can be independently embedded into external HTML pages. In the file system, this folder is `<appRoot>/_Definitions/_Widgets`
Templates	Contains definitions that allow you to take advantage of forms-based reporting, using Word, Excel, and PDF forms. Template definitions model the "fill-able" form fields contained within the source template file. In the file system, this folder is `<appRoot>/_Definitions/_Templates`
Processes	Contains definitions that provide a level of automation and contain the logic needed to perform specific tasks within an application. Process tasks can be used to perform scheduled operations, such as exporting a report to PDF format or then emailing it to a group of recipients. In the file system, this folder is `<appRoot>/_Definitions/_Processes`
Data	Contains definitions that retrieve data and provide it as a JSON stream, for use by a variety of consumers, including non-Logi applications. In the file system, this folder is `<appRoot>/_Definitions/_`

Folder/Definition	Description
	Data

From XML to HTML...

Every definition file created with Logi Studio is an XML document, which describes multiple objects we call *elements* and their *attributes*. Elements encapsulate different types of functionality and presentation; attributes enable you to customize element properties and behavior. For example, a DataLayer element retrieves data from a data source and a Chart.Pie element creates a pie chart from the data. The Chart.Pie element contains attributes to specify the height, width, color, border, and more.



Here's an example of a simple report that illustrates elements work:

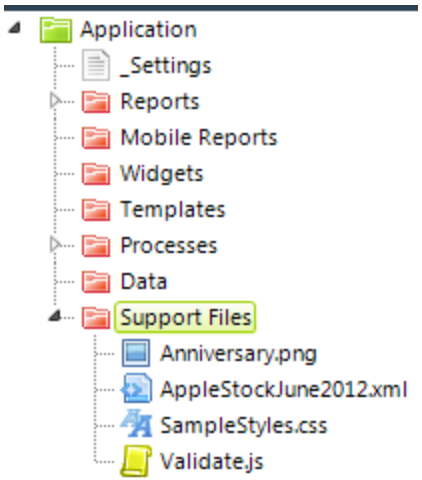
The image above left shows an example report definition, named "newReport", in Logi Studio. The report contains **Style** and **Body** elements. The Body element has a "child" element beneath it: a **Label** element, which has a **Caption** attribute.

The middle image shows the underlying XML source code for this collection of elements, as seen in Logi Studio's Source tab. You can edit this source code directly but we don't recommend it.

The Logi Server Engine processes the XML source code, generating the HTML needed to present the page in Studio's Preview tab, which is shown in the right image.

Use Support Files with a Logi Application

Studio's Application Panel also includes a folder for **Support Files**. These are files, such as images, that used by the application but are not definitions:



Typically, Logi applications include the following types of Support files:

- **Data** - Excel, CSV, XML, and other data files that are a data source for report definitions
- **HTML** - Custom HTML web content to be included within the application
- **Images** - GIF, JPG, PNG, SVG and other types of image files
- **Scripts** - JavaScript files
- **Stylesheets** - Cascading Style Sheet (CSS) files for presentation and formatting control
- **Templates** - Word, Excel and PDF form template files

You can also put just about any other file you might want to access in Support Files. In the file system, this folder is `<appRoot>/_SupportFiles`.

Application Deployment

All of the files that typically make up a Logi application reside under one folder, the *application root folder*. This folder is registered with the web server as a "virtual directory" or web application, and is almost always stored on the web server. However, it is possible to distribute the application in clustered server environments.

Publishing Logi applications, such as deploying them from a development web server to a production web server, is a simple matter of copying the application folder to the production server. Studio includes a convenient Deployment Tool that deploys applications using a variety of techniques. Updating definitions in an published application is also just a simple matter of copying them. For more information, see "Deploy a Logi Application" on page 110.

Get Started with Logi Info v23.1

The following steps guide you through the process of installing and configuring the Logi Info product and provides an overview of embedding your application.

1. Preparing your Environment

Before you begin the process of installing Logi Info, ensure that you are using a system that meets the software and hardware requirements. For more information, see System Requirements - Logi Info and Preparing to Install.

2. Install Info

To download Logi Info go to DevNet > **Support** > **Product Download**.

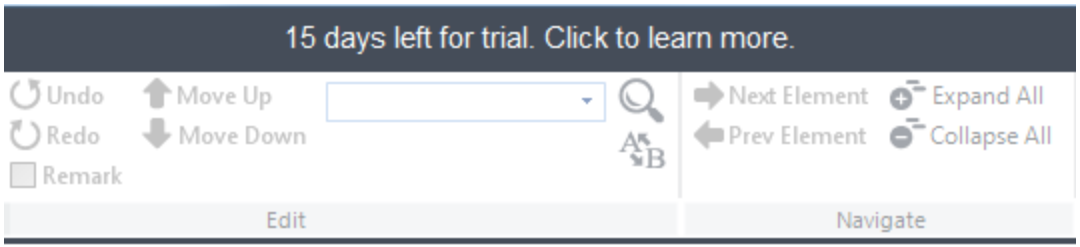
Logi Info for the Windows environment requires the .NET Framework 4.x. If not already in place, with your consent, appropriate versions of the .NET framework are installed when Logi products are installed. They are also available for free from the Microsoft Download Center. Logi Studio is not a web application; it's a .NET application and a licensed copy must be installed on the Windows computer of each developer. It allows you to create both .NET and Java-based Logi applications and reports. For more information, see Installing Logi Studio.

3. Get Your Dev License

Logi Analytics licenses are server-based rather than individual-user or concurrent-access licenses, so an unlimited number of end-users can access Logi reports through a single web server. Our licensing scheme allows you to deploy our product on one development machine and on one production server. Additional separate licenses for Studio, for additional developers, are also available. Current Logi Info versions come with a built-in **15-day trial license**. After the trial period expires, Studio and any Logi

reports you may have developed will *no longer run* without a real license.

You don't need to do anything but install the product and you can begin using it immediately. A clearly-visible display in the Studio main menu counts down the days remaining in the trial period:



Selecting the counter display will take you to a web page that offers information about purchasing a Logi Info license. Or, contact Customer Service if you want to upgrade and need a new license.

Licenses are keyed to you or your organization; they take the physical form of license files, which are assigned to a specific computer. DevNet includes a **License Management** page where you can manage your licenses, including reviewing them, assigning and un-assigning them to machines, and generating license files, at any time, without any interaction with our staff. For more detailed information about licenses, see *Product Licensing*.

4. Develop your Logi Info Application

Prior to developing your Logi application, it is imperative to read Development Best Practices to set up a good development foundation.

Elemental Development is the process of creating flexible reporting applications using predefined, XML-based objects or "elements".

The advantages of Elemental Development are:

- XML elements are reusable and encapsulate specific functionality common to most web applications.
- A hierarchical layout of elements makes it easy to manage the presentation and functionality of large web-based reports.
- Logi Studio's toolbox of elements eliminates repetitive coding, shortens development time, and minimizes errors.
- Elements can be easily added, deleted, and moved to create almost any type of report. Their **attributes** are easily configurable.

Logi Info includes our primary development tool, Logi Studio. This integrated development environment is the recommended tool for use by developers creating Logi applications. See "Introducing Logi Studio" on page 49 and Using Logi Studio.

See this video for an overview of building a simple Logi Info application. For step-by-step instructions, see Hello World! Tutorial.

Once you build your application, connect to your own data and build you own Dashboards. See Data Table Tutorial - Manual.

Here is an overview for designing Logi reports: Design a Logi Report. For customization and styling resources, see: Using Style Sheets, Themes, and Use the Theme Editor.

5. Secure your Application

Logi Info includes features that allow you to build applications that securely control access to reports and data, and prevent malicious users from adversely affecting the system. See Secure Logi Info Applications, Logi Security, and Working with Logi Security.

See this video for a demonstration on the use of Logi Security.

Logi provides the following four mechanisms: NT Authentication, Session Variable Authentication, Standard Authentication, and SecureKey Authentication. For more information, see Logi Security Scenarios.

6. Integrate with a Parent Application

Logi **SecureKey Authentication** technology provides integration for applications requiring secure access to a Logi application or embedded Logi components. It enables a "single sign-on" environment while enhancing security: user authorization is established and requests are made to Logi applications or embedded components using a special security key. For more information, see Logi SecureKey Authentication.

See this video for an introduction on SecureKey Authentication. Additionally, this video demonstrates how to set up SecureKey Authentication.

The **Embedded Reports API** provides developers with easy and agile methods for embedding Logi report components into other, non-Logi web pages. For more information, see Embedded Reports API.

See this video for a demonstration on embedding using the API. For more information on how to pass info between embedded report API and the parent application, see the Using the get() Method section in Embedding Dynamic Reports using JavaScript.

Next Steps

a. Set up your production server environment. See "Production Server Considerations" on page 111.

b. Store your code in the repository (git, code management systems). For more information, see our article on using version control with Logi Info.

c. "Deploy a Logi Application" on page 110 to your production environment. For additional information, see our Deployment Checklist and our video on deploying Logi Studio on AWS.

Launch Logi Applications

Logi reports are simply web pages and, as such, they can be accessed by entering the appropriate URL in your browser. However, there are other methods for accessing reports and related Logi application features, which are discussed here. Security is not part of this discussion, though authenticated user information can be included via session variables using the same techniques. Topics include:

Calling a Report via URL

Traditionally, a Logi report is called by entering the appropriate URL in a browser. This can be done manually by typing it right into the browser's address bar or it can be done via any valid HTTP link in a different Logi report, or in an entirely separate web page or application. The basic URL for a Logi report takes this form:

```
http://localhost/yourLogiAppName/rdPage.aspx?rdReport=mySalesReport http://www.y-
ourWebSite.com/yourLogiAppName/rdPage.aspx?rdReport=mySalesReport
```

The first example above is a URL that might appear on a development computer; the second, on a production server, where

- The **rdPage.aspx** page denotes that a Logi report definition is to be processed
- The **rdReport** argument value is the name of the Logi report definition file to be processed
- Additional request parameter name-value pairs can be included in the URL with separating ampersands (&).

More information about passing data in the URL's query string can found in our document *Pass Information*.

Specifying Report Format or Template

You can also specify a **report format** in the query string in order to output the report in different ways. This is done using the reserved word **rdReportFormat**, as follows:

```
http://localhost/yourLogiAppName/rdPage.aspx?rdReport=mySalesReport&rdReportFormat=PDF
```

When the example URL shown above is browsed, the report mySalesReport will be output into the browser as a PDF document. When you use this reserved word, ensure that you also use the rdReport name-value pair (in other words, don't rely on a report being the default report in your application and leave rdReport out of the URL). Other values for rdReportFormat include

NativeExcel, *NativeWord*, *CSV*, *HtmlExport*, *HtmlEmail*, *Excel*, *Word*, *XML*, or *GoogleSpreadsheet.* Similarly, you can use **rdTemplate** instead of rdReport to identify an Excel, Word, or PDF template definition to be run from a URL. Learn more about the reserved words that can be used in a URL for special purposes in our document *Query String Parameter Reference*.

Using a Constant in a URL

If you're using a URL within a Logi report definition to call another report, for example, by using Action.Link, you can use a **constant** and its token to provide part or all of the URL. Suppose, for example, you've assigned the first part of the URL "`http://www.yourWebSite.com`" to a constant called SiteURL, in the application's _Settings definition. The URL used to call another report or web page can then make use of that constant by using its token:

```
@Constant.SiteURL~/yourLogiAppName/rdPage.aspx?rdReport=mySalesReport
```

An arrangement like this will let you run development and production versions of an application that differ only in the value of the SiteURL constant.

Calling a Process Task via URL

If you're a **Logi Info** developer, the entry point for a Logi application does not have to be a report. Logi Info includes **Process tasks** and your application can be launched through a process definition rather than a report definition.

```
http://www.yourWebSite.com/yourLogiAppName/rdProcess.aspx?rdProcess=myProcessDef&rdTaskID=ShowMenuPage
```

The example URL shown above can be used to directly call a Process task, where

- The **rdProcess.aspx** page denotes that a Logi process definition is to be processed.
- The **rdProcess** argument value is the name of the Process definition file to use
- The **rdTaskID** argument value is the name of the Process task to run

This approach can be very useful if you want do logging, or send notifications, or accomplish any of a number of other activities that need to be done before showing the user the first report page. Use of a **Response.Link** element with a URL is also the only way to have one Process task call another Process task. As always, when working with tasks, ensure that you have a final Response.Report element that's processed at the end of the task or your user will be left seeing nothing but a blank page.

Using Dynamic Connections

Developers often want to make their datasource connection strings dynamic, based on some external criteria at launch, and this has to occur before report definitions are processed. This topic describes how to use dynamic connections.

Duplicate _Settings Definitions

If the motivation for this it to duplicate the application in several different environments (development, QA, production) but minimize changes that need to be made when moving reports through them, then the easiest approach is simply to deploy copies of all Logi application files to the different servers. Connection element **IDs** remain the same across all the _Settings definitions, but their connection string attribute values are different, as appropriate for each server. This means that report definitions that use the connections will run on any of the servers without having to be modified. Any of the other values in _Settings, such as the Application Path, can be similarly configured.

Using Session Variables

If you're using some other application that eventually calls a Logi application, you can use **session variables** to set the connection strings (or other values found in the _Settings definition) in the Logi application. Set these variables in the main application, and in your Logi application's _Settings definition, use @Session tokens for the values of your connection string, path, and other settings. When the Logi application is called, it will resolve these tokens into values and run the reports based on them.

Using Plug-ins

"Plug-ins", which are DLL's, give Logi developers the ability to **programmaticallyextend** the functionality of their Logi applications. With them, developers can dynamically change the behavior of Logi reports at runtime by accessing *request* and *session* variables and modifying report definitions. To support dynamic connection strings, a Plugin Call element is used in the _Settings

definition to retrieve the connection string (or other values) from a session variable and then it dynamically modifies the _Settings definition by inserting the connection string value. The plug-in runs on the LoadDefinition event and so alters the definition before it's processed. Learn more about plug-ins in our document *Logi Plug-ins*.

Using a Customized Global.asax File

Each Logi application includes a file named Global.asax, which is standard in ASP.NET applications. *This file is compiled in our Java products and is not accessible for customization.* As distributed, it contains only one line of code. However, event handlers can be placed here which react to ASP.NET object events, such as Application_BeginRequest and Session_Start.

Here's a sample:

```
<%@ Application Codebehind="Global.asax.vb" Inherits="rdWeb.Global" %>
<script language="VB" runat="server">

Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)
Dim rdServer As New rdServer.rdSession()
Dim myURLQStr As String = "rdProcess.aspx?rdProcess=myProcessDef&rdTaskID=mySessionStart"

' start the new session
' if session already exists, does nothing
Call rdServer.SessionStart()

' test flag to see if session has already been started
 ' if not, then do the redirect to the process task
If Session("myStartFlag") = Nothing

' save original query string to session variable for later access
Session("myQueryString") = Request.QueryString.ToString()

 ' redirect to Process task: mySessionStart
```

```
 Response.Redirect(myURL)

' set session counter
 Session("myStartFlag") = 1

End If
End Sub
</script>
```

In the example VB code above, the user is automatically redirected to a Process task called "mySessionStart" when a new session is started. The URL defined as "myURL" is a relative address, and therefore doesn't need the "http://..." part of the address. So, as another example of how to assign dynamic datasource connection strings, the code above could be adjusted to assign a connection string to a session variable and then call a Logi report.

 The two lines of code that reference the **rdServer** object *are required*, or your application will not start. **Caveat**: this example is offered here for consideration by experienced ASP.NET developers but custom code written by developers is outside the scope of our typical support offerings.

Using a Customized Web.config File

Each Logi application includes a file named Web.config, which is standard in ASP.NET applications. *This text file is accessible for customization in both .NET and Java Logi applications.*

You can add the following section to it for connection strings:

```
<connectionStrings>
<addname="myConnString"connectionString="Server=xxx; Port=nnnn; Database=xxx; uid=xxx; pwd=xxx;"/></connectionStrings>
```

Under the .NET framework there are classes and methods specifically for reading from this file. In Java, a stream reader of some kind must be used. In both cases, this can be done from within one of the plug-ins described above in order to extract the connection string data and modify the _Settings definition at runtime. The .NET framework also provides a mechanism for encrypting connection strings so that they're not held in plain text in this file. Learn more about securing connection strings in this Microsoft document.

Introducing Logi Studio

Logi Info includes our primary development tool, Logi Studio. This integrated development environment is the recommended tool for use by developers creating Logi applications.

The following topics provide a brief introduction to the features of Logi Studio:

About Logi Studio

Logi Studio is a tool designed to let developers create Logi applications as easily and efficiently as possible. The definition files that form the "source code" of a Logi application are simple text files and can be edited with any text editor; however, Logi Studio makes the development and testing process typical of programming *much* easier.

Logi Studio is not a web application; it's a .NET application and a licensed copy must be installed on the Windows computer of each developer. It allows you to create both .NET and Java-based Logi applications and reports.

All of the files that make up a Logi application, including definitions, images, style sheets, scripts, and more, can be managed within Studio. All files, with the exception of images, can be *created* and *edited* within Studio. Files can also be managed directly in the Windows file system.

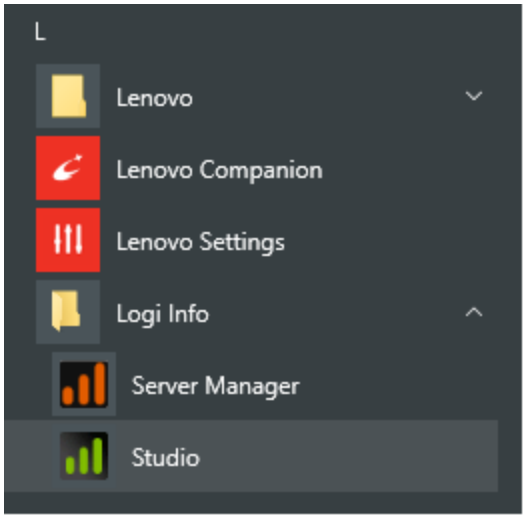
Studio includes a number of **wizards** and other tools that automate many standard development tasks, incorporates a set of "rules" that prevents you from combining incompatible objects. It also provides *Intellisense*-like features that provide code com-

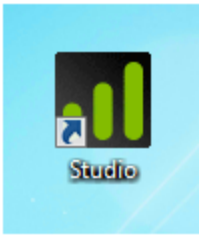
pletion options to speed development. Its element-based technique of adding and arranging objects in a hierarchical tree eliminates tedious coding and allows for code-reuse.

This topic refers to the latest version, Logi 12 Studio. A detailed user guide, *Using Logi 12 Studio*, is also available.

Launching Logi Studio

This topic demonstrates how to launch Logi Studio. When Logi Info is installed, it creates a Start menu entry for Logi Studio:





The installer (or you) may also create a Desktop shortcut, shown above, right. Click either of these to launch Logi Studio.

You can also launch Logi Studio, and open an application in it, from a Command Line window. To do this:

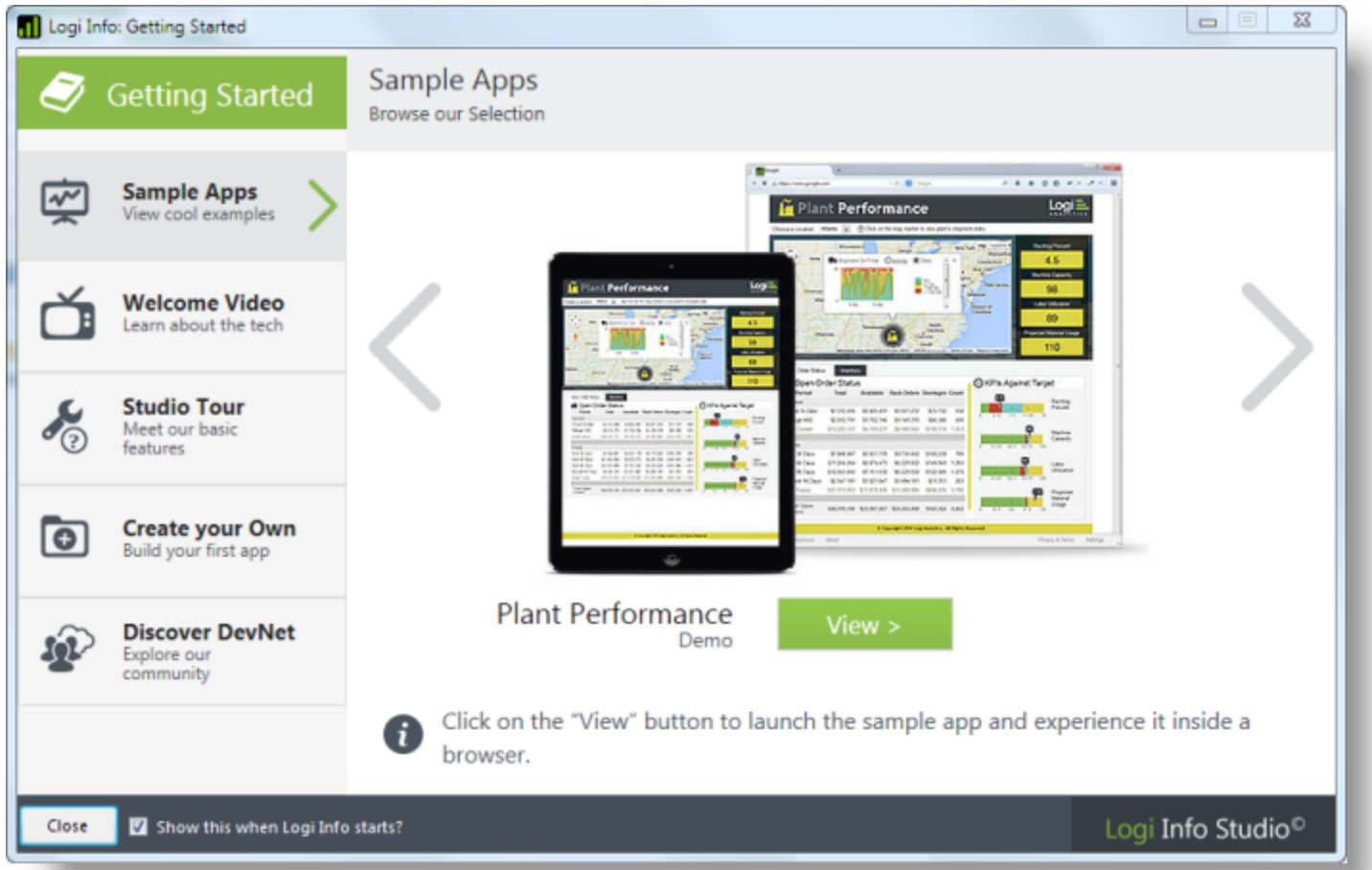
1. Open a Command Window.
2. Navigate to the Logi Info installation folder. The default is: `C:\Program Files\LogiXML IES Dev.`
3. In that folder, navigate to the `LogiStudio\bin` folder.
4. Issue the command: `LogiStudio.exe "Logi application path"` where the path is a fully-qualified path to the Logi application folder.

Logi Studio Geography

Logi Studio has a simple layout that makes it easy to develop applications and provides an array of features and tools. Learning "what's where" is the first thing you should do when beginning to work with Studio.

The Getting Started Dialog Box

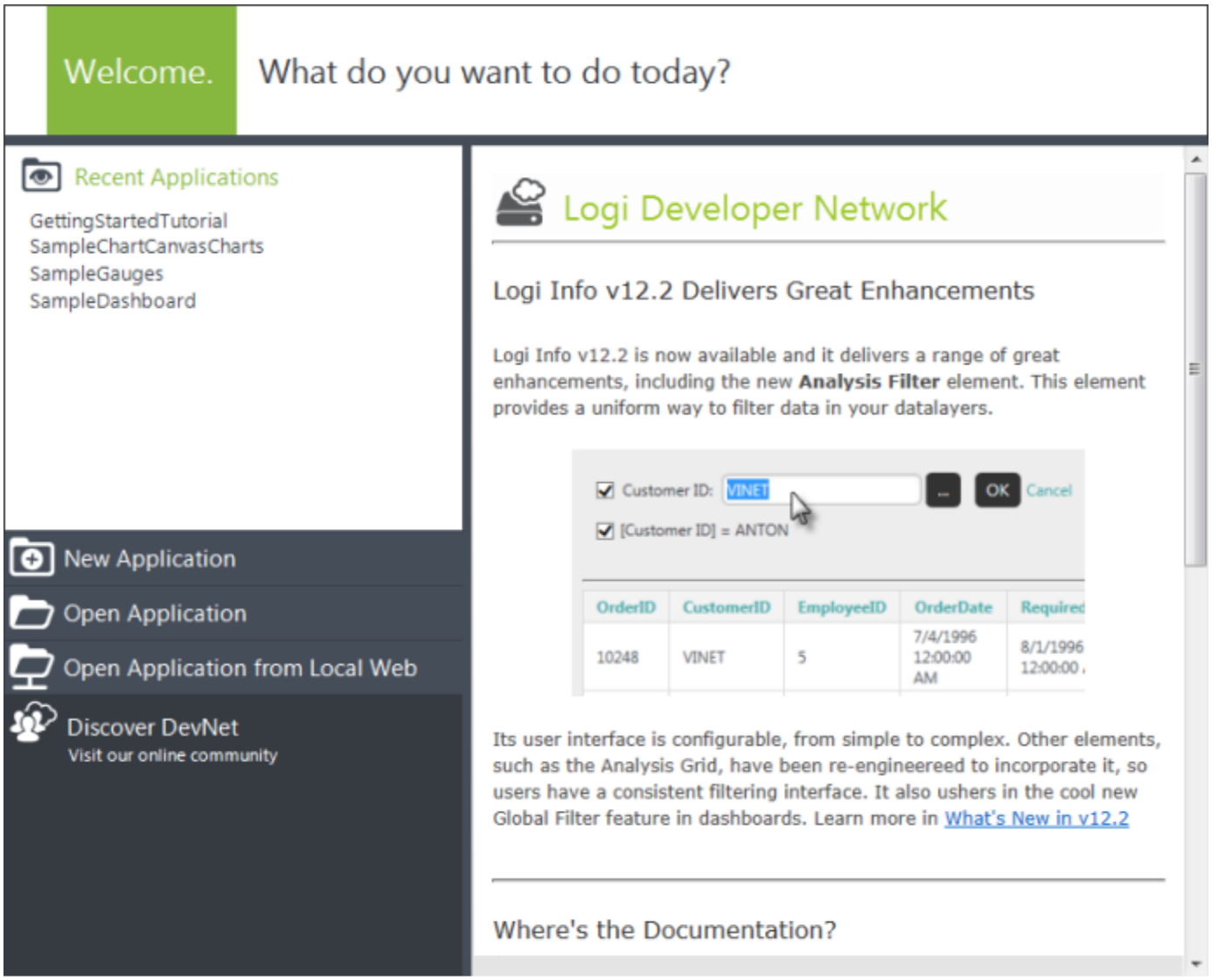
The first time you launch Studio, you'll see the **Getting Started** dialog box:



The dialog box, shown above, contains everything you need to quickly become familiar and productive with Studio. You should definitely explore all of its contents if you're a Logi Info newbie. You can hide it when you no longer need it.

The Welcome Panel

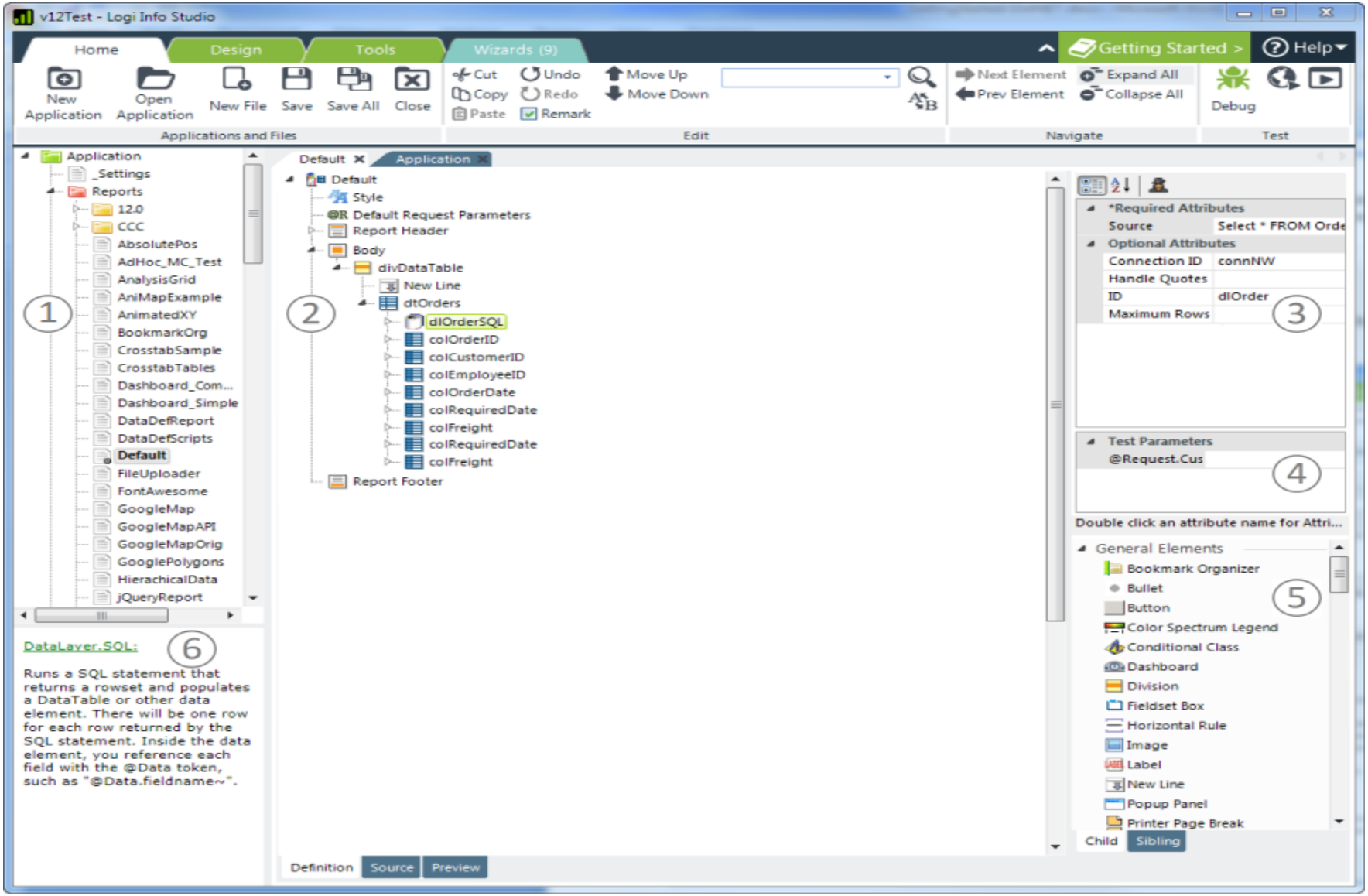
With the Getting Started dialog box hidden, you'll see the **Welcome panel** when you start Studio and whenever there's no application open.


Welcome.
What do you want to do today?
Recent Applications
GettingStartedTutorial
SampleChartCanvasCharts
SampleGauges
SampleDashboard
New Application
Open Application
Open Application from Local Web
Discover DevNet
Visit our online community
Logi Developer Network
Logi Info v12.2 Delivers Great Enhancements
Logi Info v12.2 is now available and it delivers a range of great enhancements, including the new Analysis Filter element. This element provides a uniform way to filter data in your datalayers.
Customer ID: VINET
OK
Cancel
[Customer ID] = ANTON
OrderID
CustomerID
EmployeeID
OrderDate
10248
VINET
5
7/4/1996 12:00:00 AM
8/1/1996 12:00:00
Its user interface is configurable, from simple to complex. Other elements, such as the Analysis Grid, have been re-engineereed to incorporate it, so users have a consistent filtering interface. It also ushers in the cool new Global Filter feature in dashboards. Learn more in What's New in v12.2
Where's the Documentation?

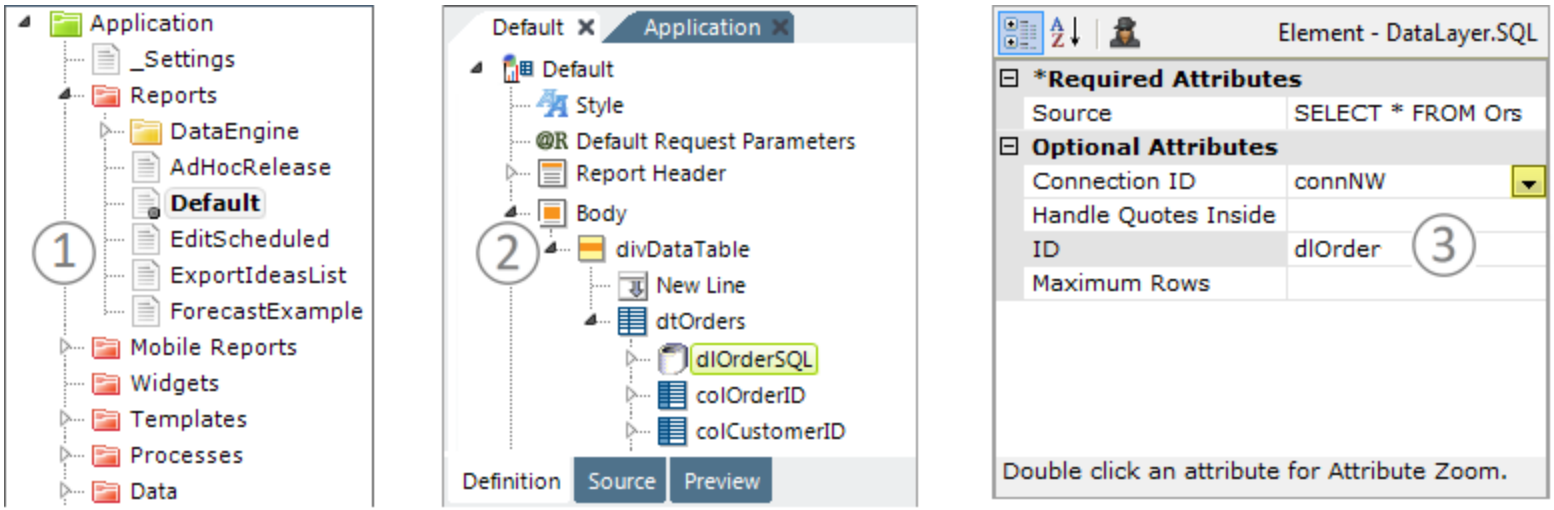
The Welcome panel provides quick links to recently-opened applications, links that open applications using a variety of methods, and a link to DevNet, our online community. In addition, you'll see some useful content, from DevNet, displayed in a scrolling area.

Working on an Application

When you open an application in Studio and begin editing a report definition, it looks like this:



Logi Studio uses **six standard panels**, which can be resized to suit your needs:



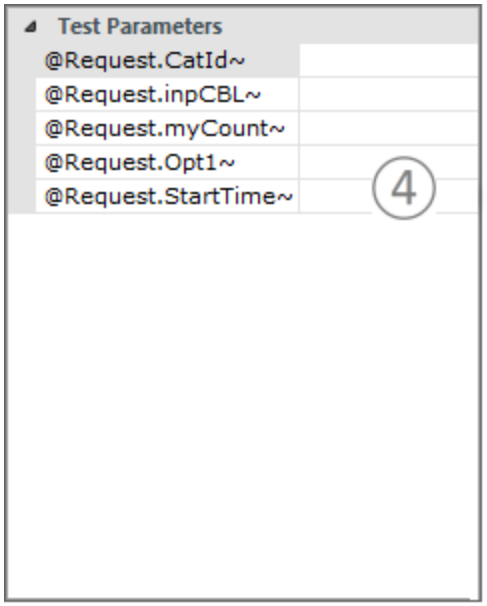
1. The **Application** panel groups all of the files that make up a Logi application together into one file management tree.

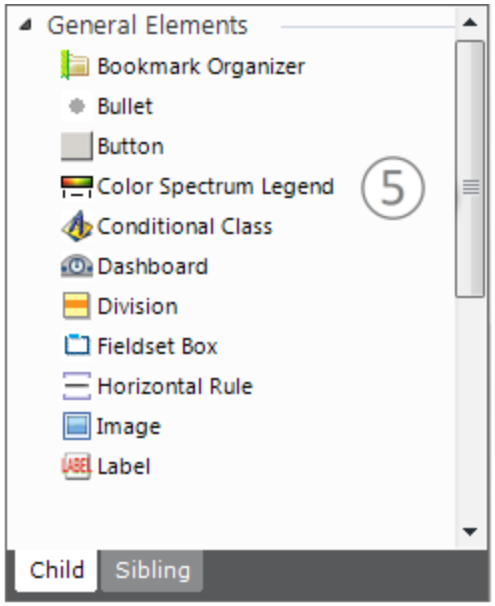
   A *Filter Definitions* control is available at the top of the Application panel for filtering definitions in all categories at once. Clear the control to remove all filters.

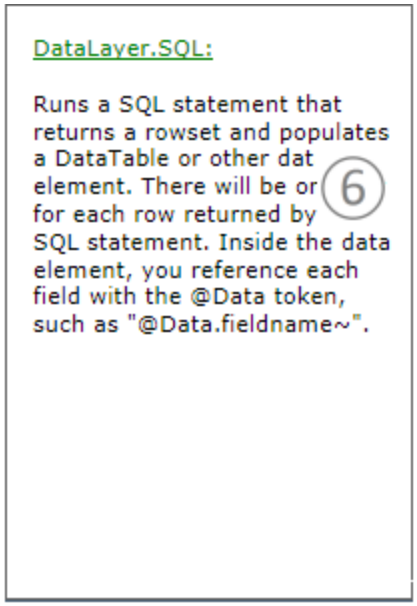
2. The **Workspace** panel is where files are edited. Tabs at the top are displayed for each file that's opened, as well as the Welcome and Application pages. Different editors are displayed within the tab pages depending on the file type being edited (report definition, style sheet, XML data, etc.). Report definitions (shown) display the elements that make up a report, in a hierarchical Element Tree.

   When editing report definitions, tabs at the bottom are displayed for switching between views of the Element Tree (the Definition), the underlying XML source code (Source), and a mini-browser preview of your work (Preview). When the Preview or Source tab is selected, the Element Toolbox, Attributes, and Test Parameters panels are automatically hidden, allowing the full width of the Workspace panel to be used.

3. The **Attributes** panel is where developers enter element attribute values. Attributes can be sorted **Alphabetically** or by **Category** and the **Attribute Spy** feature can be invoked. Attribute names can be double-clicked to open a "zoom" window for easier data entry. For some attributes, valid choices, in a selection list, can be summoned by clicking a down-arrow or mini-browse button in their value field.







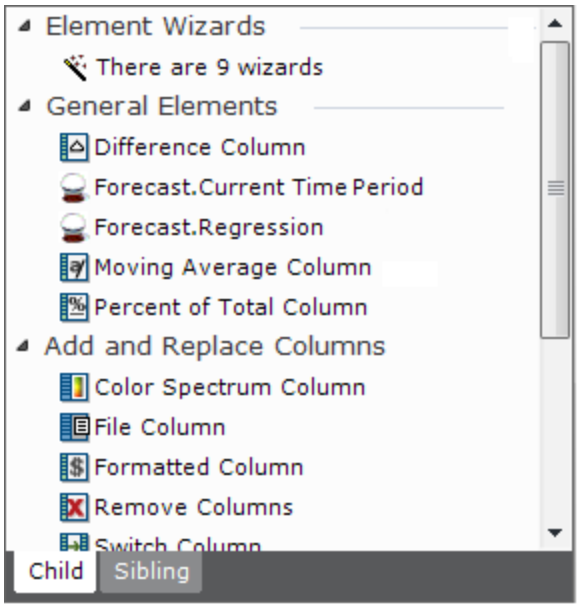
4. The **Test Parameters** panel allows you to insert values as for parameters that are normally passed to the page. This panel is only visible if the report expects parameters and is not displayed until a report has been run or previewed once. Test values entered here are applied to a definition viewed *inside Studio using Preview*, but are *not* applied if the definition is viewed in a browser.
5. When an element is selected in the Element Tree in the Workspace panel, the **Element Toolbox** panel provides a list of context-appropriate **wizards** and child elements that can be added to the Element Tree. Tabs at the bottom filter the element selection between child and sibling elements.

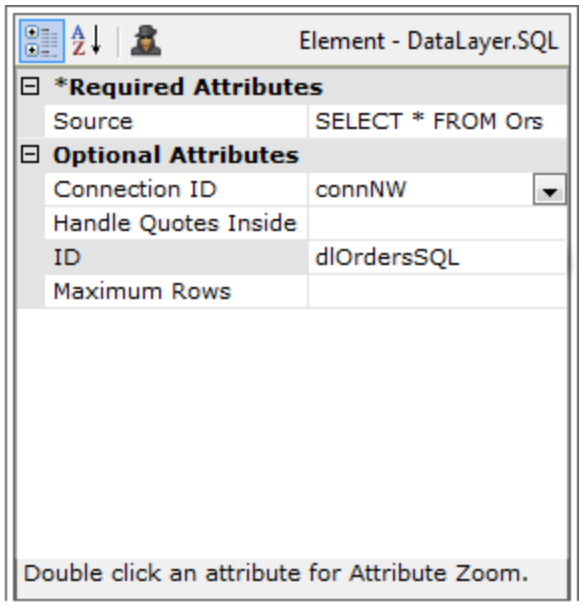
A *Filter Elements* control is available at the top of the Element Toolbox panel for filtering elements in all categories at once. Clear the control to remove all filters.

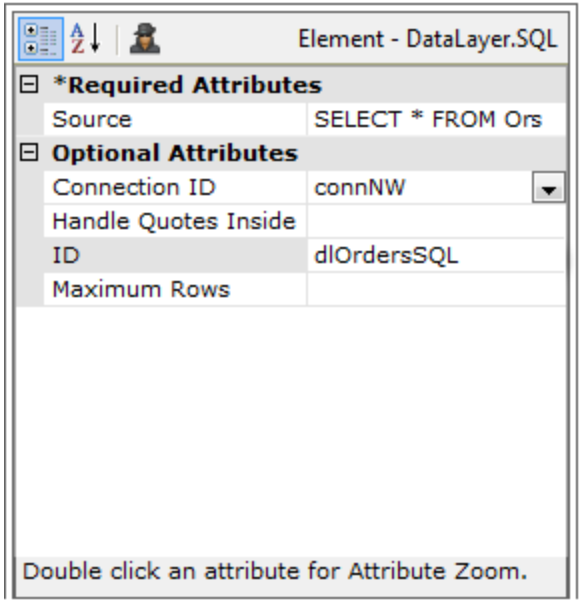
6. The **Information** panel includes quick help text; the element or topic name is a link to more information on DevNet.

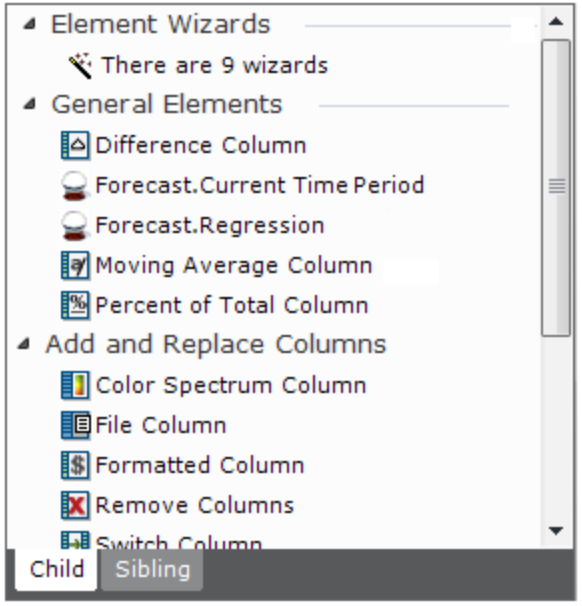
Panel Arrangement

The Attribute and Element Toolbox panels can be re-arranged to suit your preference, using the Tools → View menu options:





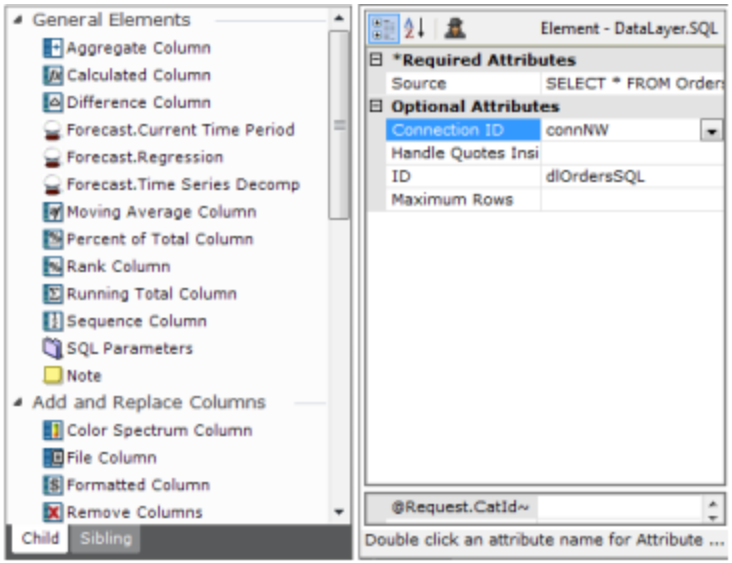




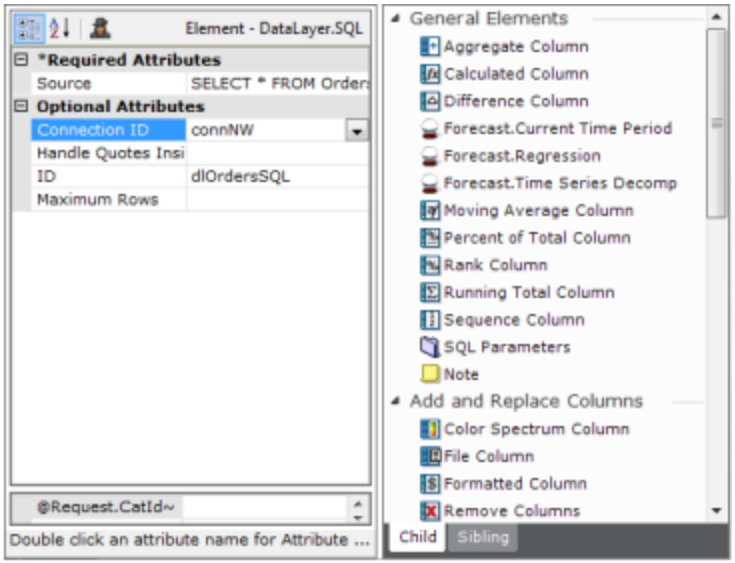
Elements then Attributes

Attributes then Elements

In a *vertical* layout, the order of the Element Toolbox and Attributes/Test Parameters panels can be specified, as shown above.



Elements then Attributes

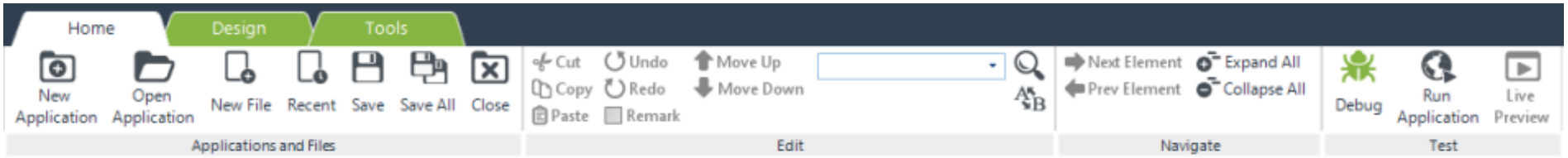


Attributes then Elements

Or, in the optional *horizontal* layout, the panels can be arranged in the order you desire, as shown above.

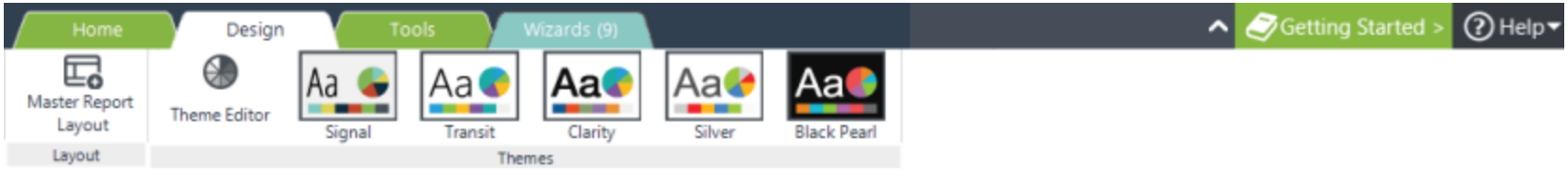
Logi Studio Ribbon Menu

Studio operations can be controlled using the **Ribbon Menu** at the top of the screen. Here are its tabs and major items:

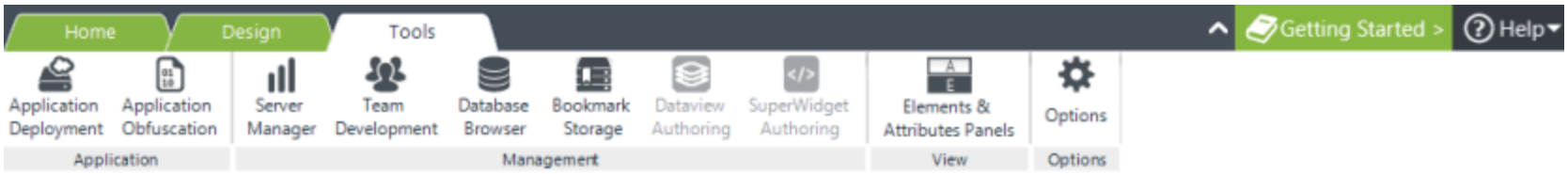


Feature	Description
New Application	Create a new application with the assistance of the New Application wizard.
Open Application	Open an existing application from the File Explorer, the recent apps list, or a list of apps registered with the local web server.
New File	Create a new file by selecting a file category.
Recent	Re-open a recently closed file.
Save, Save All, Close	Save the file in the current Workspace editor tab, save all edited files, and close the current application, with a prompt to save any unsaved files.
Edit Actions, Remark, Move Up/Down	Standard editing actions including Cut, Copy, Paste and more. Comment and uncomment elements and rearrange their order.

Feature	Description
Search/Replace	Search current, or all, definitions and supporting text files, and replace text, if desired.
Element Navigation, Expand/Collapse	Navigate to previous or next element in the Element Tree navigation history. Expand or collapse all child elements beneath the selected element.
Test Actions	Control debugging, and preview a definition in a browser or in a separate "live" window.

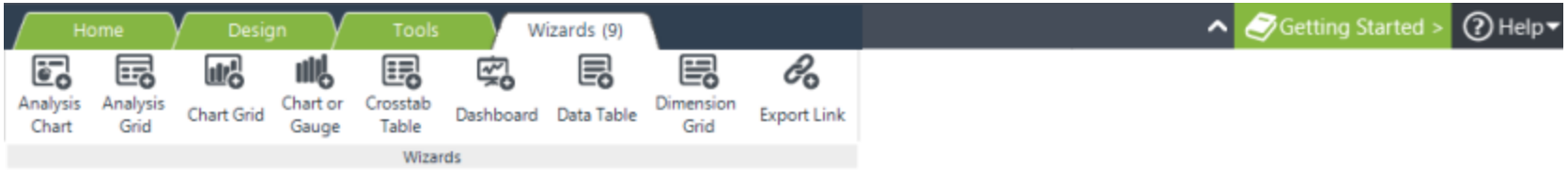


Feature	Description
Master Report Layout	Runs the Master Report Layout wizard, which lets you design the layout of a master report template.
Theme Editor	Runs the Theme Editor wizard, which lets you change Theme colors, typography, etc.
Themes	Apply the selected Theme to the current report, or to the entire application.



Feature	Description
Application Deployment	Deploy the application to other servers using the Application Deployment Tool.
Application Obfuscation	Render definitions unreadable by humans to prevent tampering and theft using the Application Obfuscation Tool.
Server Manager	View registered applications and change their Logi Engine versions individually or in bulk.
Team Development	Manage built-in Logi source code control with file check-in/out and change history.
Database Browser	View schema, stored procedures, and basic data samples from databases with connections in the _ Settings file.
Bookmark Storage	Create a SQL database table for Bookmark storage and migrate Bookmarks stored in the file system into it.
Dataview Authoring	Create, edit, and manage Dataviews - representations of data that can be used and re-used throughout an application. *Menu item is only visible when Discovery v3.x is installed.* *Deprecated in Info v12.6+.*

Feature	Description
SuperWidget Authoring	Create, edit, and manage client-side components, such as charts and Dashboards, to create single-page applications that make use of Dataviews. *Menu item is only visible when Discovery v3.x is installed.* *Deprecated in Info v12.6+.*
Elements/Attributes Panel	Specify the arrangement and order of the Element Toolbox and Attributes panels in Studio.
Options	Specify Studio options including file encoding and definition editor colors.



The Wizards tab appears when an element that has supporting wizards is selected in the Workspace panel. The number of wizards available varies, depending on the selected element.

Logi Studio General Features

Logi Studio provides development convenience and efficiency. It includes many features added in direct response to requests from our developer community. General features include:

- **Wizards** - More than 50 wizards are provided to help you create applications and visualizations, and to accomplish other activities.
- **Multiple-Document Editor** - The Workspace panel allows multiple files of different file types to be opened simultaneously, including report and process definitions, style sheets, script files, and XML data files. Elements and attribute values can be easily copied between definitions.
- **Support File Editing** - Studio's internal support file editor provides improved editing and intelligent formatting of the source material based on file type. Features include line numbering, collapsible code regions, and syntax-specific high-lighting.
- **Global Search and Replace** - The editor's Find and Replace feature supports current document and cross-document searching using a variety of criteria.
- **Attribute Spy** - This feature displays in the element tree, next to all relevant elements, their values for a selected attribute. For example, this feature allows you to quickly see which style class has been assigned to each element.
- **Tree-based Element Selection** - Child elements can be selected and added to the element tree either from the Element Toolbox panel or by right-clicking a parent element and selecting from the popup menu that appears.
- **Intelligent Code and Token Completion** - The task of entering tokens in attribute values is simplified by this feature, which dynamically presents selection lists for token types and appropriate identifiers in a drop-down list.
- **Attribute Value Selection Lists** - Values can be selected from drop-down lists for attributes with discrete sets of value options, such as True/False, Percent/Pixels, etc.
- **Color Picker Tool** - Attribute color values can be selected from a "color picker" tool, which can be opened from the attrib-ute value.
- **Theme Editor** - An editor that allows you to create custom Themes, based on the provided standard Themes, by selecting colors, typography, and more in an easy-to-use interface.

- **Test Parameters Panel** - Parameters used in previews and tests remain accessible for easy use in the Test Parameters panel.
- **Element Copy/Paste and Drag/Drop** - Multiple elements can be selected at once in the element tree and copied, moved, or deleted.
- **Query Tools** - The SQL Query Builder is a separate dialog box that allows queries to be created from graphic table representations, using drag-and-drop techniques. SQL statements can be extended across multiple lines, allowing more complex statements. In addition, statements can be tested by selecting them and pressing F5; the operation will be limited to just the selected portions of the query statement.
- **Attribute Display Order** - Element attributes can be viewed listed either in functional groups or sorted alphabetically.
- **Database Browser** - This handy utility launches from the toolbar and allows browsing of the structure and data in connected databases. It's also a non-modal window, so it can be kept open and referred to it while working with elements and attributes.
- **Style Class Selector** - This utility, shown as the first option in a the drop-down list of available classes for any Class attribute, appears as a dialog box and displays available classes and previews their effects as each is selected. Classes can also be edited right in place in the dialog box.
- **Deployment Tool** - Allows you to easily deploy Logi applications by copying their files to local or shared network drives, or using one of three flavors of FTP (regular, SSL, or SSH).

Detailed information about how to use Studio is available in *Using Logi 12 Studio*.

Add-on Modules

Add-on Modules for Logi Info are designed to extend Logi Info functionality. They enable special elements in the Logi Engine and may include complete Logi applications.

Applications included in the Add-on Module come with all their definitions and support files and you can examine and customize them, if desired. They may also include features that make it easy for you to "brand" them with your own logo, company name, etc. to make them your own.

Like all Logi applications, Add-on Module applications can be distributed without paying a licensing fee to Logi. However, these applications use very advanced techniques; taking them apart and copying portions of them into other applications may not produce the desired results.

 Note the following important considerations:

- Modules may require a specific Logi Info release and may *not* be backwardly compatible.
- Module components may work with a restricted set of data sources; see the descriptions below for details.

Support and updates for Add-on Modules are covered by existing Logi Info Support and Maintenance Plans. If you've purchased our plans, then Add-on Modules you subsequently purchase are also covered.

To purchase an add-on module, contact your Logi sales representative directly or via SalesTeam@LogiAnalytics.com.

Topics related to add-on modules include:

- "Available Modules" on the next page
- "Release Pairings" on page 72

Available Modules

Add-on Modules for Logi Info provide developers with extended functionality, complete solutions, or other benefits.

The following modules are currently available:

Module	Description
Self-Service Reporting Module (SSRM)	The Self-Service Reporting Module enables specialized Logi Info elements, and includes **InfoGo**, a complete Logi application. "Out of the box", the InfoGo application uses these elements to allow your users to create, save, and share their own data visualizations, Dashboards, and reports. You can use InfoGo as a stand-alone application or embed it in other applications. Its full source code is provided and you can easily brand the application as your own. See *Install the Self-Service Reporting Module*for installation information.
Discovery Module (DM)	The Discovery Module enables specialized elements that allow you to embed the **ThinkSpace** into a Logi Info application. The ThinkSpace is Logi's unique drag 'n' drop interface in which users can create data relationships and visualizations by using their mouse to make onscreen connections. Intuitive algorithms suggest the most appropriate visualizations based on the data selected. This modules allows developers to provide an incredible amount of functionality with minimum effort. The Discovery Module includes a new technology, **Logi Services**, which includes features advanced data retrieval and management features that work with background services to provide a rich, modern development experience and a robust, highly-interactive user experience. See *Installing the Discovery Module - Windows*and *Installing the Discovery Module - Linux*for installation information for the latest version.
Logi Predict Module (LPM)	The Logi Predict Module brings predictive analytics to your end-users. It enables specialized Logi Info elements and includes a complete Logi application. "Out of the box", the Logi Predict application uses these elements to let your users analyze historical or transactional data and make statistical predictions about current data. You

Module	Description
	embed it in other applications. Its full source code is provided and you can easily brand the application as your own. Go to the *Introducing Logi Predict* topic for Logi Predict for more information.

Requirements include Logi Info v12.1+, Chrome v26+ or IE 10+ or Firefox v20 or Safari v6+.

Release Pairings

Modules may require a specific Logi Info release and may *not* be backwardly compatible.

Logi Info, SSRM, and Discovery

The table below shows which product versions we recommend using together:

Release Date	Logi Info Version	SSRM Version	DM Version
31 Mar 2023	23.1	14.1.612, 12.8.592 compatible	3.2
17 Jan 2023	14.2 SP4	14.1.612, 12.8.592 compatible	3.2
30 Nov 2022	14.2 SP3	14.1.612, 12.8.592 compatible	3.2
31 Oct 2022	14.2 SP2	14.1.612, 12.8.592 compatible	3.2
09 Sep 2022	14.2 SP1	14.1.612, 12.8.592 compatible	3.2
26 Aug 2022	14.1 SP4	14.1.612, 12.8.592, 12.7.348 compatible	3.2
July 2022	14.2	14.1.612, 12.8.592 compatible	3.2
27 May 2022	14.1 SP3	14.1.612, 12.8.592, 12.7.348 compatible	3.2

Release Date	Logi Info Version	SSRM Version	DM Version
28 April 2022	14.1 SP2	14.1.612, 12.8.592, 12.7.348 compatible	3.2
24 Mar 2022	14.1 SP1	14.1.612, 12.8.592, 12.7.348 compatible	3.2
25 Feb 2022	14.0 SP3	12.8.592, 12.7.348 compatible	3.2
27 Jan 2022	14.1	14.1.612, 12.8.592, 12.7.348 compatible	3.2
29 Oct 2021	14.0 SP2	12.8.592, 12.7.348 compatible	3.2
28 Sep 2021	12.8 SP4	12.8.592, 12.7.348 compatible	3.2
31 Aug 2021	14.0 SP1	12.8.592, 12.7.348 compatible	3.2
15 July 2021	12.8 SP3	12.8.592, 12.7.348 compatible	3.2
25 June 2021	14.0	12.8.592, 12.7.348 compatible	3.2
31 Mar 2021	12.8 SP2	12.8.592, 12.7.348 compatible	3.2
29 Jan 2021	12.8 SP1	12.8.592, 12.7.348 compatible	3.2

Release Date	Logi Info Version	SSRM Version	DM Version
21 Dec 2020	12.7 SP10	12.7.348	3.2
11 Nov 2020	12.8	12.8.592	3.2
16 Sep 2020	12.7 SP9	12.7.348	3.2
31 July 2020	12.7 SP8	12.7.348	3.2
10 June 2020	12.7 SP7	12.7.348	3.2
18 May 2020	12.7 SP6	12.7.348	3.2
24 April 2020	12.7 SP5	12.7.348	3.2
19 Mar 2020	12.7 SP4	12.7.348	3.2
23 Jan 2020	12.7 SP3	12.7.348	3.2
5 Dec 2019	12.7 SP2	12.7.348	3.2
8 Oct 2019	12.7 SP1	12.7.348	3.2
27 Aug 2019	12.7	12.7.348	3.2
25 Jun 2019	12.6 SP6	12.6.308	3.2

Release Date	Logi Info Version	SSRM Version	DM Version
17 May 2019	12.6 SP5	12.6.308	3.2
19 Apr 2019	12.6 SP4	12.6.308	3.2
29 Mar 2019	12.6 SP3	12.6.308	3.1
05 Mar 2019	12.6 SP2	12.6.308	3.1
18 Jan 2019	12.6 SP1	12.6.289	3.1
03 Dec 2018	12.5 SP10	12.5.272	3.1
15 Nov 2018	12.6	12.6	3.1
15 Oct 2018	12.5 SP9	12.5.272	3.1
05 Oct 2018	12.5 SP8	12.5.272	3.1
23 Aug 2018	12.5 SP7	12.5.272	3.1, 3.0 SP3 compatible
1 Aug 2018	12.5 SP6	12.5.272	3.0 SP3
20 Jul 2018	12.5 SP5	12.5.272	3.0 SP3
16 Jul 2018	12.5 SP5	12.5.272	3.0 SP2

Release Date	Logi Info Version	SSRM Version	DM Version
29 May 2018	12.5 SP4	12.5.266	3.0 SP2
21 May 2018	12.5 SP3	12.5.266	3.0 SP2
30 Mar 2018	12.5 SP2	12.5.266	3.0 SP1
2 Feb 2018	12.2 SP17	12.2.244	2.3 SP2
29 Jan 2018	12.5 SP1	12.5.266	3.0.1830, 2.3 SP2 compatible
9 Jan 2018	12.2 SP16	12.2.244	2.3 SP2

Logi Info and Logi Predict

The table below shows which product versions we recommend using together:

Release Date	Logi Info Version	LPM Version
Mar 2019	12.6.121-PA	3.0
Sep 2018	12.5.240-PA	2.1

Release Date	Logi Info Version	LPM Version
Jun 2018	12.5.211-PA	2.0
Feb 2018	12.5.139-PA	1.2
Mar 2018	12.5.162-PA	1.3

For more information, see the Release Notes for each Logi Info or add-on module version.

The _Settings Definition

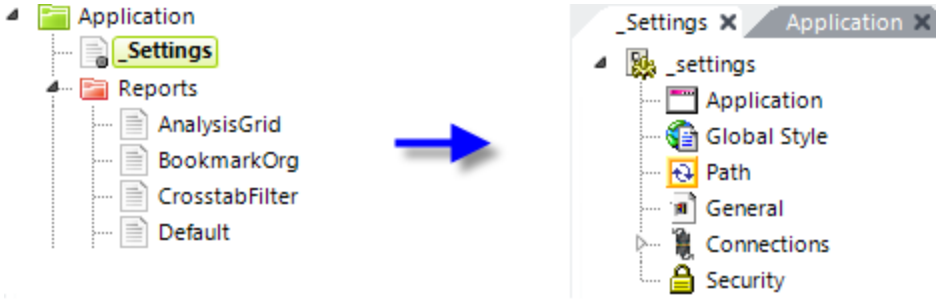
The _Settings definition contains global configuration settings for your Logi application. This topic introduces the _Settings definition and the elements available for use in the definition.

The following topics describe the purpose of the elements that can be used in this definition and provide links to more detailed information, when available:

- Application Element
- Connections Element
- Constants Element
- Event Logging Element
- External Definitions Element
- File To Database Mapping Element
- General Element
- Global Chart Export
- Global Style Element
- Globalization Element
- Java Session Copying Element
- Path Element
- Security Element
- Session Timeout Element
- Startup Process Element
- Team Development Element
- The Wait Panel Element

About the _Settings Definition

The _Settings definition (*yourLogiApp*_Definitions_Settings.lgx) contains the global configuration settings for your Logi application.



It includes a variety of unique elements, some required, some optional, that provide overall control of the application. The required and most-commonly-used elements are shown above.

The definition includes information like database connection strings and security information that may need to be changed when deploying the application from development to production. For this reason, care should be taken when considering copying or overwriting a _Settings definition file.

The _Settings.lgx file of a Logi Info application is expected to be in a nested location under the root folder of the application, specifically in: `<application root>/_Definitions`.

Combining _Settings definition files for multiple applications or "centralizing" them elsewhere is not possible. A _Settings definition can be obfuscated, using the Obfuscation tool in Logi Studio, a common technique used by developers to protect their software products from theft, tampering, and reverse-engineering. For more information, see *Obfuscating Definitions*.

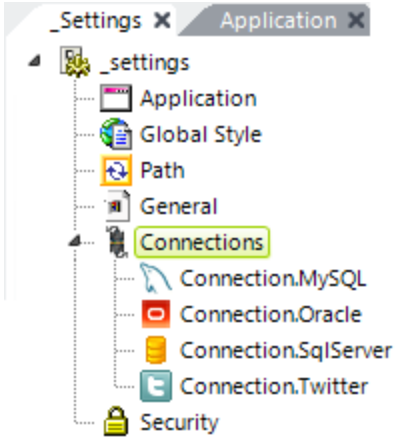
Application Element

(Required) The **Application** element determines the general characteristics of the application. Its attributes are:

Attribute	Description
Caption	Specifies the text to be displayed in browser tabs or title bar, if no captions have been set in other, individual report definitions.
Default Report	Specifies the report definition to run when one is not specified in an rdReport parameter in the requesting URL. When this attribute is blank, the system looks for a definition named "Default".

Connections Element

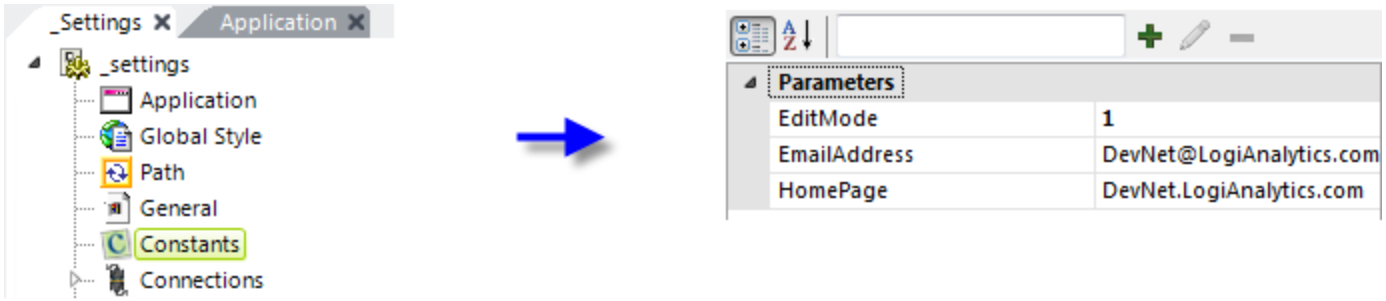
The **Connections** element is container for **Connection** elements, which provide the connections to datasources. A connection established here in _Settings is available for use by any number of report definitions.



Connection elements are available for vendor-specific and generic datasources, including databases, online services, and data files. In most cases, they simplify the process of connecting to data. More information about them is available in *Datasource Connections*.

Constants Element

Constants are global, literal values that will not change and that can be used in all definitions. They're created in _Settings, using the **Constants** element.

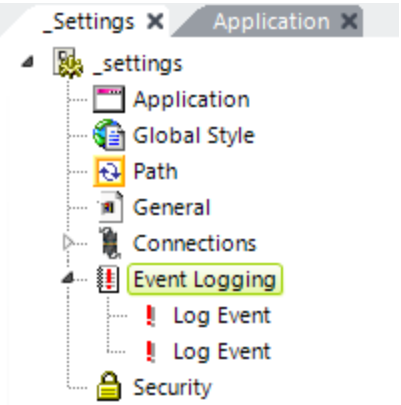


Constants are created, as shown above, as a parameter list and are accessed using @Constant tokens. For example, the value of the first constant in the list shown above would be available using `@Constant.EditMode~`. For more information on @Constant tokens, see *Token Types*.

The use of constants provides a way to have values referenced throughout your application while only defining them in a single place, which makes maintenance very easy. Constants are often used for values that might change between development and production environments.

Event Logging Element

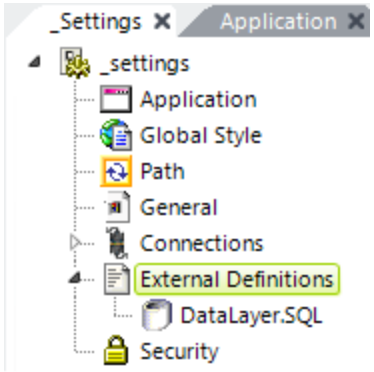
The **Event Logging** element enables logging of various events such as session starts, authentication, and database queries. Logging works in conjunction with a Process definition file that usually handles inserts into a logging database table.



The Event Logging element is a parent for one or more Log Event elements, each of which logs a specific event. More information about event logging is available in *Event Logging*.

External Definitions Element

Application report definition files are normally stored in the `yourLogiApp\ _Definitions\ _Reports` folder. However, developers can instead specify definitions that come from an external data source, such as a database or a web service, using the **External Definitions** element.



As shown above, the External Definitions element works with its own datalayer, which retrieves the definition data when the application runs. At runtime, the presence of an External Definitions element alerts the **Logi Server Engine**to automatically include the datalayer's data in the Engine's inventory of report definitions. When the report is called, the Engine will read the definition from the datalayer rather than from the web server's file system and render it.

The stored definition data should be in two columns:

ReportID (varchar/nvarchar(50)) - which contains the report definition name (without .lgx), and
Definition (varchar/nvarchar(max)) - which contains the complete XML source of the report definition.

When building the data, the XML could be copied, for example, right from Studio's Workspace, Source tab, and inserted into the datasource. All values are case-sensitive.

To use an external definition, reference its ReportID value (the literal text - "BalanceSheet" - not as an @Data token) in the **Report Definition File** attribute of elements such as **Target.Report**.

Why would you want to store your definitions in a database? It's a bit more complicated to update definitions in this arrangement and it isn't necessarily for everyone. However, it does allow centralized storage of report definitions, integrates with Content Management Systems, provides portability in virtualized-server environments, and works well in web server farms, so it definitely has its uses.

File To Database Mapping Element

The **File To Database Mapping** element enables storing Bookmark, Gallery, and Save files in database tables. Storing this information in a database instead of the file system can make it easier to manage system backups, create cloud deployments, and more. For complete information about the use of this element, see *Storing Bookmark, Gallery, and Save Files in a Database*.

The element's attributes now support tokens.

General Element

(Required) The General element is the catch-all for a variety of configuration settings and has two child elements, Global Chart Export and Wait Panel. Its attributes include.

Attribute	Description
Bookmark Admin Editor Security Right ID	Specifies (when using Logi Security) a comma-separated list of Security Right IDs. Users that have one of these rights can edit bookmarks that have been shared with them.
Bookmark Collection Default	Specifies the default Bookmark Collection attribute value, which is used in a number of elements. Using this attribute helps ensure consistent values for all bookmark collections. This value is required when sharing bookmarks with the Bookmark Organizer element. For more information, see *Bookmarks*. If using Logi Security, you can specify a personal collection for each user by including the `@Function.Username~` token here.
Bookmark Folder Location	Specifies the fully-qualified path and folder name where bookmarks will be stored. Bookmarks make it easy for users to build a menu of reports with saved input parameters and can save user configurations of Analysis Grids, Analysis Charts, OLAP Grids, and Dimension Grids. Bookmarks can also be stored in a database, see *Storing Bookmark, Gallery, and Save Files in a Database*
Cookie Expiration	Specifies the number of days a cookie created by the application will last. The cookie expiration date is calculated from the present day. The default value is *30 days*. You may set the value to *Session* (the equi-

Attribute	Description
	valent of *0* days) and cookies will expire when the session ends.
Cookie Path	Specifies where cookies created by the application will be stored. By default, cookies are stored at the web server level , where they are available to any application on the server. If you want a customized arrangement instead, the Cookie Path attribute allows you to specify a single storage location for all cookies, using a URL. Enter a single slash ("/") for this attribute value to have all cookies be stored at the web server (domain) level, for example "myServer/", or enter a more complex URL for other locations in the Logi application. The web server domain is assumed, so don't include the domain in the value and begin it with a slash ("/").
Data Cache Folder	Specifies the fully-qualified path and folder name for storing temporary cache files. Ensure that the ASP.NET system account has full access rights to the folder. You can use a token, such as `@Function.AppPhysicalPath~`, specify part of the path. Temporary files in this folder are only read by the application and it does not need to be accessible by the end users via HTTP. The default value is the application's `rdDataCache` folder. For a web farm environment, specify a network folder and change the application's identity from ASP.NET to a network account.
Debugger Security Right ID	Specifies a comma-separated list of one or more Security Right IDs that enable debugging for the current user and session. For more information, see *Debug Reports*.
Debugger Style	Specifies which style of debugger should be used. Options include:

Attribute	Description
	NoDetail - Prevents any debugger information from appearing when there's an error, so the user cannot see sensitive system security information. This is the default value. Use this option when deploying to production systems. *ErrorDetail* - When an error occurs, displays a detailed error message. *DebuggerLinks* - Shows the "Show the Debugger Trace Page" icon or similar link at the bottom every report page. Clicking it displays the Debugger Trace Page. Use this option only during development. For more information, see *Debug Reports*. *DebuggerLinksNoData* - Shows the "Show the Debugger Trace Page" icon or similar link at the bottom of every page, but the Debugger Trace Page will not include links to the data. Use this option only during development. *Redirect* - When an error occurs, redirects to a your custom error page. You must specify the URL to redirect to in the **Redirect Error URL** attribute.
Disable Metadata Builder	Specifies whether access to the Web Metadata Builder tool will be disabled for*everyone*. The default value is *False*. Access to the Web Metadata Builder can be restricted more selectively by assigning values to the Security element's Metadata Admin Security Right IDs attribute.
Doctype Declaration	Adds a Document Type Declaration to the top of the HTML output, which alters the way browsers react to style attributes and tags in HTML and Cascading Style Sheets (CSS). Follow the link above for information

Attribute	Description
	about each standard option. For more information, see *Doctype Declarations*. The default is *Html5*, which produces this declaration: `<!DOCTYPE HTML>` You can type-in any value that begins with "<!DOCTYPE" if you want to specify a custom option.
Error Log Location	Specifies an optional,fully-qualified path and folder name where Error Log files will be written when an error occurs, if the Log Errors attribute is set to *True.* Ensure that the account used to run the application has full File Access rights to the folder. The default value is `\rdErrorLog` in the application folder. Tokens can be used here, if desired: `@Function.AppPhysicalPath~\MyErrorLogFolder` For use with a Web Farm, specify a shared network folder and change the application's identity to a network account.
Filter Case Sensitivity	Specifies the whether the Analysis Filter element will be case-sensitive when matching text values. Also affects super-elements that utilize the Analysis Filter internally, such as the Analysis Grid and Dashboard. When set to *Sensitive*, values match only when the case of the text matches. When left blank or set to *Insensitive*, case does not matter - so, for example, "abc" will match "Abc". v12.2 SP3 - The option, *DataSourceCollation*, is also available and applies only when using ActiveSQL data-sources. This option sets the filter's case sensitivity to that of the underlying column or database and results in improved filtering performance, especially for columns that are indexed.
IE Compatibility	Specifies the inclusion of the `X-UA-Compatibility` meta tag in the HTML output. In some cases, especially

Attribute	Description
	when it detects that it's browsing an Intranet, Internet Explorer automatically reverts to IE7 compatibility mode, disabling newer browser and HTML features. This attribute helps ensure that Internet Explorer runs in the appropriate mode/version. The default value is *IE=edge*, which has IE run using the latest features of the IE browser. Set to *None* to prevent output of the tag.
iFrame Embedding Host Restrictions	Specifies options for the X-Frame-Options directive in the HTML response. This can be used to restrict how embedded Logi app pages may be framed by a parent application. Specify values such as *SAMEORIGIN*, *DENY,* or a free form option such as a single host name value or a list of comma-separated host names. When set to *SAMEORIGIN*, the browser will block rendering only if the origin of the parent application browsing context is different than the origin of the embedded Logi report. When set to *DENY*, browsers will prevent the embedded Logi report from rendering because it is in a frame. When set to a free form value, the browser will block rendering only if the origin of the parent application browsing context is different than the host name value specified. If a list of comma-separated host names is provided, the parent application needs to supply its own origin information as part of the link-params attribute in the HTML page mark-up, using the custom link parameter "forOrigin". For example, if the value `http://hostname1,http://hostname2` is specified then the parent HTML should include the following link parameter as part of the data-linkParams "{'forOrigin','http://hostname2'}". The engine will verify that the *forOrigin* value matches one of the values of the accepted host names and set the X-Frame-Options directive to allow embedding.
Input Value	Specifies the delimiter character to be used in the Request tokens generated by User Input elements, such

Attribute	Description
Delimiter	as Input Select List, that allow multiple value selection. The default value is a comma - ",".
License File Location	Specifies the fully-qualified path and name for the folder containing the Logi product license file. Normally, the license file is located in the root folder of each Logi application. However, it may be desirable to have a common location for a single license file for all Logi applications on the server. This attribute specifies such a common folder. If you using this attribute, you must ensure that all Logi applications run under an account with file access rights to files in the specified folder.
License User Tracking Folder	Specifies a fully-qualified path name for the folder where "tracking" files will be stored. Certain special types of product licenses may log anonymous authentication information. The default location is the `rdUserLog` folder, under the application's root folder. In some cases, it may be desirable to set a common location on a network for multiple web servers. Ensure that all applications run under an account with file access rights to create and update files in the specified folder.
Log Errors	Specifies whether the Debugger Trace Report page files will be written, as HTML files, to the folder specified in the Error Log Location attribute when an error occurs. In order to log the same level of detail found in a typical Debugger Trace Report page (Debugger Style attribute set to *DebuggerLinks*), the Debugger Style attribute must be set to *Redirect* and you must specify a Redirect Error URL attribute value.

Attribute	Description
	If Debugger Style is set to *ErrorDetail* or *NoDetail*, then the log pages will only include the level of detail provided by the corresponding error messages. The files, while typically small in size, are not managed automatically and the system admin will need to do it manually.
OEM Distribution License	Specifies the code for a valid OEM Distribution License, which allows Logi Info applications to run on a server that does not otherwise have a license key installed. This is especially useful for XCOPY deployments as the Logi Info installer program does not have to be run on the server. Copy the entire XML contents of the OEM license file into this attribute.
Redirect Error URL	Required if the Debugger Style attribute is set to *Redirect*. Specifies the URL for a custom error page to be displayed when an error occurs. You may specify a relative or absolute URL, and the custom error page must be an .aspx page. When an error occurs, before executing the redirect to this URL, the system sets a session variable named `rdLastErrorMessage`. If desired, you can use this variable in the custom error page to display the actual error message.
Report Author Upload Folder Override*	Specifies the default value for the Report Author element's Uploaded File Folder attribute. This can be a fully-qualified file system path, or one relative to the web application. Tokens such as @Function.UserName~ can be used here to individualize the folder. If the folder is *outside* of the application folder, at runtime the image files will be automatically copied to the rdDownload folder prior to rendering them to the client.

Attribute	Description
	* Prior to v12.2 SP3, this attribute's name was Report Author Upload Folder.
Repository Friendly	Specifies whether or not to format definition source code for use with 3rd-party source code repositories, such as GIT and TFS. This value can be set manually here or by using Studio Options. When a new application is created using Studio's New Application wizard, this value is set to *True*. For more information about its effects, see *Using Logi 12 Studio*.
Script Source Debugger Style	Specifies the level of debugging support for scripting in the Debugger Trace Page. Options include: *Always* - Includes a "View Script" link in the trace page when a script is invoked, if the script is more than one line. If it's only one line, then the script line itself is displayed. *OnError* - Includes the "View Script" link described above only appears when an error has occurred. *None* - No link or script source is included. This is the default value.
Script Value Cache Count	Specifies the maximum number of script formula results to be cached. Caching eliminates the need for a script to be evaluated by the scripting engine repeatedly. The default value is *500*. To disable caching, set the value to *0*.
Scripting Language	For .NET applications, specifies the scripting language for attributes that support scripting. The default is *JavaScript*, which is recommended for new applications. VBScript is still supported but is now deprecated. For Java applications, the scripting language defaults to *JavaScript* regardless of this attribute setting.

Attribute	Description
~~SQL Injection Guard~~ *This attribute was deprecated - see Description*.	 With the deprecation of this feature, and in general, we *highly recommend* that you prevent SQL Injection attacks during development by passing all values into SQL statements as parameters using the **SQL Parameters** element, or by using SQL Stored Procedures. The syntax to be used for parameters in queries vary by database. For MS SQL Server, for example, question mark placeholders are used for parameters: `SELECT * FROM Orders WHERE OrderID = ?` Injection attacks can potentially perform database inserts, updates, and deletions. A good practice for guarding against such attack-spawned actions is to connect to the database with an account that does not have privileges for those operations, if they aren't needed.
Studio Element Seeker Port	The Element Seeker locates elements, in Studio, that have been clicked in the browser and it's enabled when this value is set. This attribute specifies the TCP port number used for communication between Studio and the browser. For convenience, use Studio's ribbon menu Debug item to enable and disable the Element Seeker.

Global Chart Export

The **Global Chart Export** element is a child of the General element. It automatically enables *all* Chart Canvas Chart and Gauge elements to make their visualizations available as a .PNG image to be downloaded and saved.

For more information about using this feature, see *Export Chart Canvas Charts*.

Global Style Element

The **Global Style** element allows you to specify a Theme (see *Working with Themes*) and/or a separate style sheet (see *Style Sheets*) that will apply to the entire application. This provides a single point for appearance management across the application. Values entered here can be overridden in individual reports, if desired, by using the **Style** element in their definitions.

Globalization Element

The **Globalization** element allows you to set application-wide characteristics related to *Internationalization and Localization* , and culture. Its attributes include:

Attribute	Description
Default Input Date Format	Specifies a date format that will be used when an **Input Date** element doesn't have its own Format attribute set. For more information, see *Format Data*.
Default Input Date Reformat	Specifies a date format that will be used when an **Input Date** element doesn't have its own Input Date Reformat attribute set. For more information, see *Format Data*. When this attribute is specified, Input Date element values are reformatted when they're referenced as @Request tokens and the format of @Date tokens are also changed. For more information, see *Token Reference*.
Default Input Time Format	Specifies a time format that will be used when an **Input Time** element doesn't have its own Format attribute set. For more information, see *Format Data*.
Default Input Time Reformat	Specifies a time format that will be used when an **Input Time** element doesn't have its own Input Time Reformat attribute set. For more information, see *Format Data*.
First Day Of	Specifies the first day of the fiscal year, which is used when calculating dates and quarters for fiscal calendars.

Attribute	Description
Fiscal Year	The value must be in the format `MM/DD`. For example, if the fiscal year begins on March 5, enter *03/05*. The default value is *01/01*.
First Day Of Week	Specifies how the first day of the week is determined for @Date tokens that return first and last days of the week, such as `@Date.ThisWeekStart~` and `@Date.ThisWeekEnd~`. This attribute value is a number representing the days of the week, where: *0* = Sunday, *1* = Monday, *2* = Tuesday, *3* = Wednesday, *4* = Thursday, *5* = Friday, *6* = Saturday
Input Reformat Culture	This attribute has been deprecated.
Java Font Folder	For Java applications only: Specifies the fully-qualified path and folder name where TrueType fonts are located for PDF exports. The folder is required for PDF exports with fonts that contain characters that are not in the ISO 8859_1 character set, such as Arabic, Cyrillic, and Korean language characters. Depending on your Java environment, you may even need to point the Logi application to the location of your regular fonts in order to ensure that they'll be used. A typical value for a Windows installation will be something like `C:\Windows\Fonts`. For Linux, it could be something like `/usr/local/share/fonts/ttfonts`.
Metric Prefix String	Specifies comma-separated list of custom formatting characters for use with Metric Prefix formatting. Setting Format attributes to "mp" formats numbers with a "metric prefix". For example, to format the value `1,234,567` as `$1.23M`, the Format attribute value is `$#.00mp`.

Attribute	Description
	Supported metric prefixes are in the range of `1000^-6` to `1000^6` and the default string of formatting characters is `a,f,p,n,µ,m,k,M,G,T,P,E`. You can replace this string with your own comma-separated string of characters to represent this range.
User Culture	Specifies a custom value that overrides the "culture" value obtained from the user's browser at runtime. Tokens, such as @Session or @Request, may be used here. Culture values are specified as a string, such as *en-US*. For a list of cultures, see your browser's options. For example, in Internet Explorer, select Tools → Internet Options → Languages → Add.

Java Session Copying Element

Java-based Logi applications and non-Logi Java applications may maintain their session variables in different ways. When these two kinds of applications are integrated, they may need to "copy" session variables so they can be shared between them. Logi session variables are accessed via @Session tokens, while standard Java application session variables are accessed via JSP or, in a Java program, via `javax.servlet.http.HttpSession`.

The **Java Session Copying** element, which is ignored in .NET Logi applications, enables session variable copying between the two kinds of Java applications. It has four attributes that let you control which variables are copied, which optimizes performance. The "Include" attributes are processed first, then the "Exclude" attributes. The attributes include:

Attribute	Description
Copy From Java Exclude	Specifies a comma-separated list of regular expressions that identify the Java application session variables that *will not* be copied to the Logi Info application session space. Do not use session variable names that contain spaces.
Copy From Java Include	Specifies a comma-separated list of regular expressions that identify the Java application session variables that *will* be copied to the Logi Info application session space. Do not use session variable names that contain spaces.
Copy To Java Exclude	Specifies a comma-separated list of regular expressions that identify the Logi Info application session variables that *will not* be copied to the Java application session space. Do not use session variable names that contain spaces.
Copy From Java	Specifies a comma-separated list of regular expressions that identify the Logi Info application session variables that *will* be copied to the Java application session space. Do not use session variable names that contain spaces.

Attribute	Description
Exclude	

In Logi Info versions prior to v11.0.115, almost all session variables were automatically copied, which didn't always produce the best performance. However, if you want to restore this behavior, you can copy the following element source code and paste it into your _Settings definition:

```
<JavaSessionCopying CopyToJavaInclude="^" CopyToJavaExclude="DebugFile,-bUsesSort$,-tra$,-xmlDef$,-Xsl$,-rdDef$"
CopyFromJavaInclude="^"
CopyFromJavaExclude="^rd,^dt" />
```

Path Element

(Required) The **Path** element tells the Logi Engine where to find definition files and Studio what the application URL is. Its attributes are:

Attribute	Description
Alternative Definition Folder	Specifies a fully-qualified path and folder name for a custom location for definition files. This can be useful for web farm configurations or in other cases in which an alternate definition location is useful. The alternate folder must preserve the standard folder hierarchy by containing a _Definitions folder with its traditional subfolders for _Reports, _Processes, _Widgets and/or _Templates. Definition files will be used from the alternate folder if they're not found in the application's regular _Definitions folder. Do not include the _Settings definition file in the alternate _Definitions folder. This attribute value can be a physical or network path. If it's a network path, ensure that the ASPNET process account has network file access rights. This is often done by enabling impersonation in `Web.config`. You may need to set the Anonymous account to a network account.
Application Path	Specifies the URL of the application, which is used by Studio to preview and test the application. This is set automatically by the New Application Wizard. If you move or rename your application, you will need to reset this attribute.

Security Element

The **Security** element configures the application's security features and enables Logi Security. Flexible authentication and authorization elements provide access control at many levels: application, report, process, element and data row. This is discussed in detail in several documents, beginning with Working with Logi Security.

Session Timeout Element

The **Session Timeout** element allows you to ensure that a user's session will not end prematurely, either automatically or by prompting the user, and is discussed in detail in *Managing Session Timeout*.

Startup Process Element

The **Startup Process** element lets you run a task *automatically* when a user session begins (i.e., when the application is browsed by a user for the first time). This allows you to set variables, run procedures, and accomplish any other "startup" work you might want to do, and then redirect the browser to a report definition.

Multiple Startup Process elements can be used; they'll be processed in the order in which they're listed in the definition.

When the Startup Process element's **First Session Only** attribute equals *True*, the Processes Task runs only for the server's first session. This enables startup processing to occur when the application starts, rather than when any subsequent session starts.

Any Request variables in the query string used to call the application will automatically be available as Request tokens in the process task called with this element. For example, `@Request.rdReport~` will contain the name of the report definition.

The startup task runs *before* Connection elements are processed. This can be especially useful if you want to set session variables based on user identity or security parameters in the task, then use them as @Session tokens in Connection element attributes to create dynamic connections.

Unlike most process tasks, you *do not* need to use a **Response** element at the end of this task. Once the startup task is finished, the complete original URL will be sent to the browser again, automatically, so you don' t need to worry about specifying a report definition or forwarding request parameters. It couldn't be easier.

Of course, if you *want* to do some evaluation in the task, with Procedure.If, for example, and then redirect to some *other* report definition, you can use Response and Target elements to do that.

More information about Process tasks can be found in *Process Tasks*.

Team Development Element

The **Team Development** element enables file locking and file history, making it easier to work on Logi applications as a team.

File locking lets users lock and unlock files and see who is using what file. This helps prevent multiple developers from working on the same file at the same time.

File history retains copies of all file revisions. You can list and view old versions and files. Old versions can be "rolled back", thus making them the current version.

For detailed information about Studio's Team Development features, which work with this element, see *Using Logi 12 Studio*.

Wait Panel Element

The **Wait Panel** element is a child of the General element and is automatically added to all new applications. When present in the _Settings definition, it's global in effect andcauses a Wait Panel to be enabled anywhere one can be used, providing a default for the entire application. The Wait Panel element attribute values become the defaults for the entire application.

If a "local"Wait Panel element exists in a report definition, the attributes of that element will be used when displaying a Wait Panel. However, any empty local element attribute values, such as Caption, Class, or Caption Class, will default to the values set in the global Wait Panel element in _Settings.

For all Wait Panel elements (except in the scenario in the previous paragraph):

- If there is no specific Caption attribute value specified, the value defaults to "Please Wait...".
- If there is no specific Class value specified, the valuedefaults to "rdThemeWaitPanel".
- If there is no specific Caption Class value specified, the valuedefaults to "rdThemeWaitCaption".

Deploy a Logi Application

You've developed and tested your Logi application and it's working perfectly, running on the "localhost" server in your development computer. Now you're ready to deploy it onto your production server.

The following topics explain how to deploy a Logi application:

- Production Server Considerations
- Using Studio's Info Application Deployment Tool
- Alternate Method: Manually Copying Your Info Application
- Deploying a Logi Services Database
- Deploying to Cloud-based Platforms
- Post-Deployment Server Configuration

If you have not already done so, we recommend reading *System Requirements - Logi Info* before proceeding.

Production Server Considerations

Before deploying your Logi application, review the following to ensure that your production server is ready to accept it:

- On a development machine you generally install both **Logi Studio** and **Logi Info Web Server**. You may also install both of them on a production server but it's not required. Just install Logi Info Web Server, which will make our **Server Manager** tool available on the production server as well.
- Check to make sure you have the proper combination of JDK with your .NET Framework version (4.x). Oracle has changed its Java usage policies - see Java Usage Policy for important information.
- Check to make sure your login credentials for accessing the production datasource have not been changed in your _Settings definition after deployment.
- Are there any corporate conventions for file or folder locations on the production server that you need to observe (such as, "all applications must be installed on the D: drive, not the C: drive")?
- Are you allowed to *remotely access* the production server using Remote Desktop, Citrix, or some other method? (This can be extremely useful when you need to run the web server's management tool or other tools.)
- If deploying with Studio's Deployment tool over a network, by copying or via a form of FTP, do you have the necessary *security credentials* to access the destination?
- If deploying by manually copying files, can you map a *network drive* that's been "shared" on the production server?

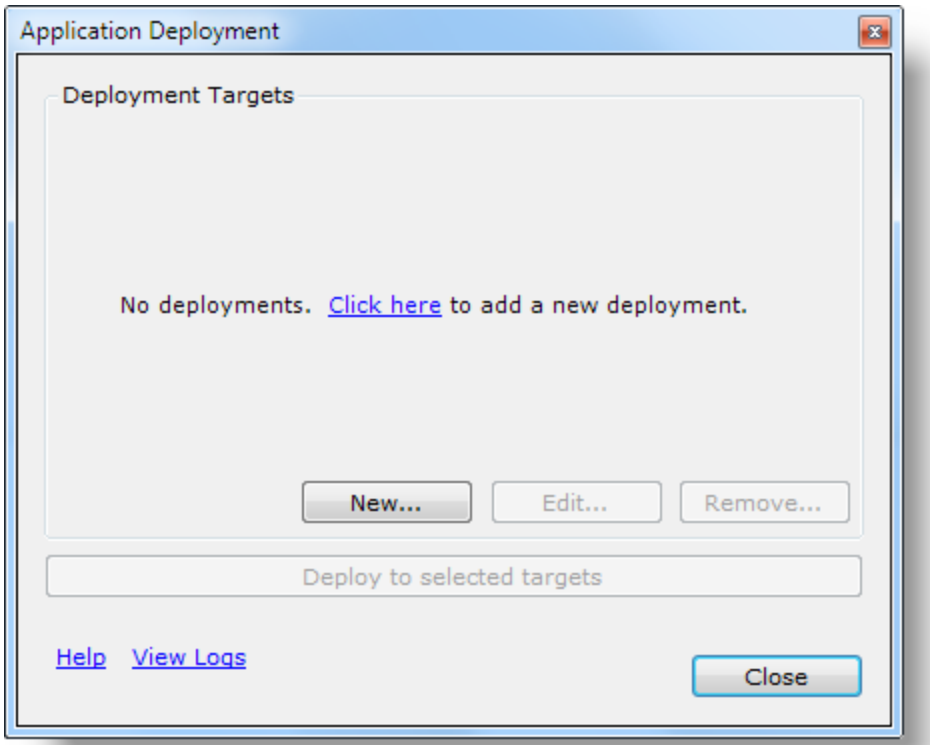
Once these items have been reviewed and addressed as necessary, you're ready to proceed with deploying your Logi application.

Using Studio's Info Application Deployment Tool

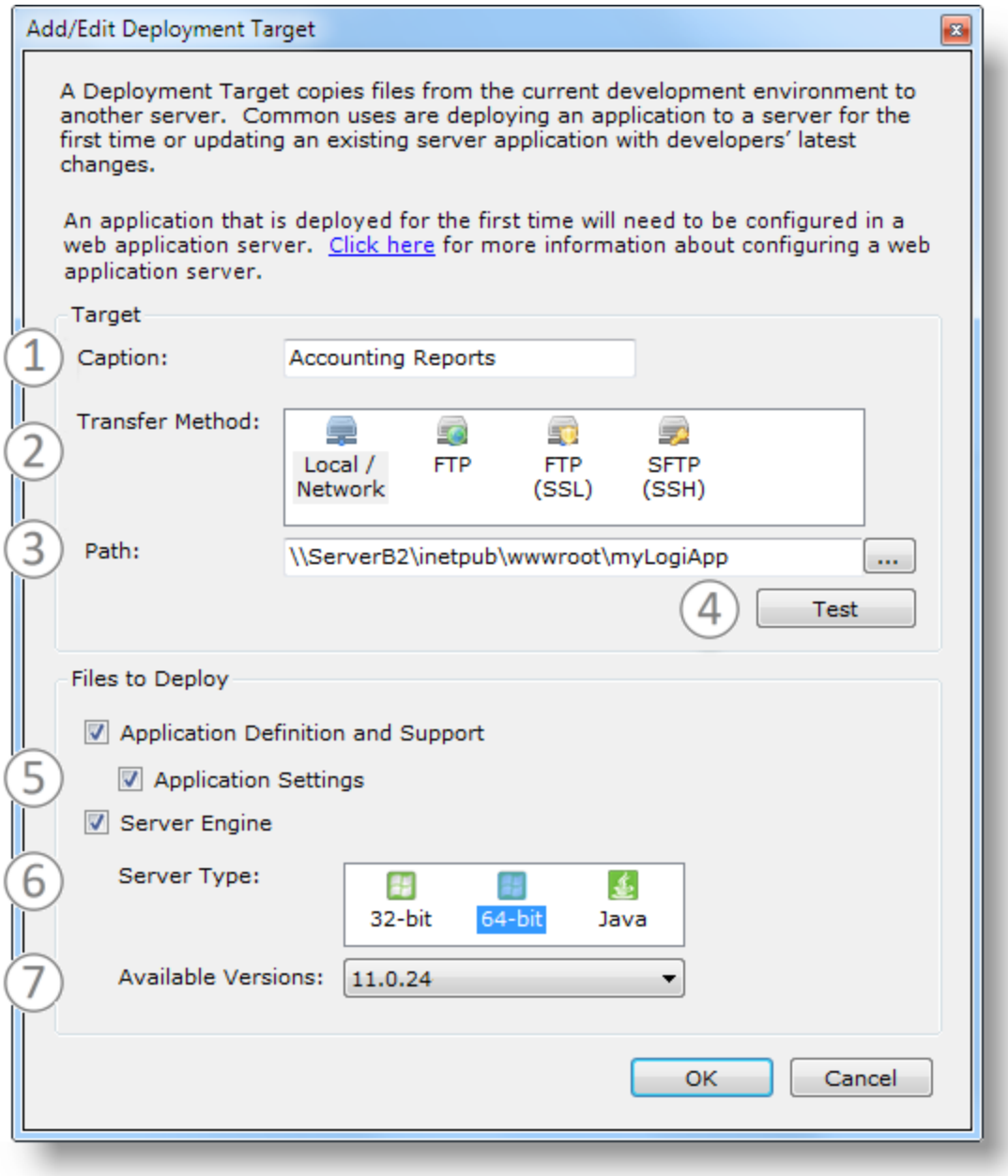
Logi Studio's **Application Deployment Tool** is easy to use and allows you to set up reusable deployment operations. The tool provides a great deal of flexibility, allowing you to develop, for example, on a 32-bit platform and deploy to a 64-bit platform, or to develop on a Windows platform and deploy to a Java platform.

The Deployment tool is capable of copying all or part of your application to the production server, using either file system commands (to local or shared network folders) or one of three flavors of FTP (standard, SSL, or SSH). In Logi Studio, the details of each operation are called a "deployment target" and they can be saved for later reuse.

The Application Deployment tool is opened in Studio by clicking its menu item, in the main menu's **Tools** tab, shown above.



If no deployments have been defined, the dialog box shown above appears. Click the link or **New...** to define your first deployment target.


Add/Edit Deployment Target
A Deployment Target copies files from the current development environment to another server. Common uses are deploying an application to a server for the first time or updating an existing server application with developers' latest changes.
An application that is deployed for the first time will need to be configured in a web application server. Click here for more information about configuring a web application server.
Target
1
Caption:
Accounting Reports
2
Transfer Method:
Local / Network
FTP
FTP (SSL)
SFTP (SSH)
3
Path:
\\ServerB2\inetpub\wwwroot\myLogiApp
...
4
Test
Files to Deploy
Application Definition and Support
5
Application Settings
Server Engine
6
Server Type:
32-bit
64-bit
Java
7
Available Versions:
11.0.24
OK
Cancel

The Add/Edit Deployment Target dialog box contains the following:

1. **Caption -** An arbitrary name you give the deployment, for future reference.
   Note: This is also used as the Deployment Log File name, so it has to adhere to the OS *file naming standards*. For example, in Windows, these characters are invalid in a file name: * | / \ : " < > ? and should not be used here.
2. **Transfer Method -** Copy to Local or Network drive, FTP, FTP (SSL), or FTP (SSH).
3. **Path** - If the "Local/Network" transfer method is selected, enter a UNC file path to a destination app folder on the production server. For example:

   `\\myWebServer\myLogiApp`

   If the destination app folder is not on the local machine, it must be *shared* on the network.

   If one of the FTP transfer methods is selected, enter an FTP protocol URL to the destination app folder and arguments. Examples of these include:

   For FTP & FTP SSL: `ftp://myWebServer/myLogiApp, ftp://myWebServer.com:8082/projects/myLogiApp`

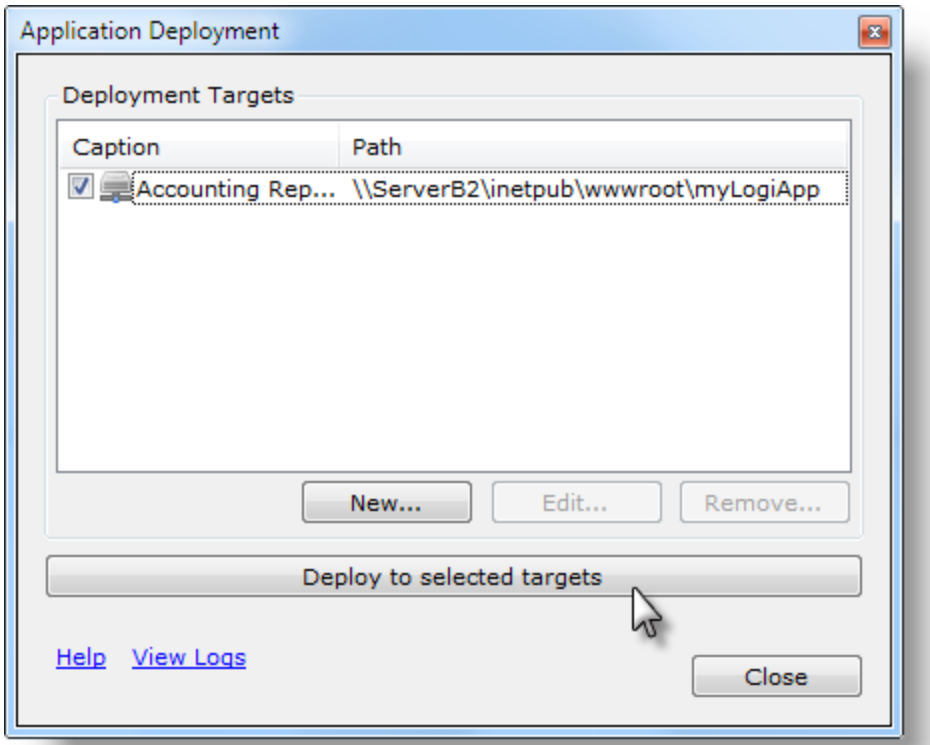
   For FTP SSH: `sftp://test.myServer.local//opt/tomcat/webapps/myLogiApp`
   Note use of *two* forward slashes after server name. Also you'll have to create the target application folder and grant permissions, as described in "Alternate Method: Manually Copying Your Info Application" on page 126, because the application is not being deployed to a user's home directory.

   If necessary, you'll be prompted to supply a user ID and a password; do not include them in the URL.
4. **Test Button** - This button can be used to test the connectivity to the destination.
5. **Files to Deploy** - Check which types of files are to be deployed. *Files are overwritten at destination without warning!*

6. **Server Type** - If "Server Engine" has been selected above, the server type options for the deployment are shown (scroll down to see additional types, such as Java). Select the server type.
7. **Available Versions** - If "Server Engine" has been selected, the available Logi Info versions installed on the development machine are listed. An effective way to perform an application upgrade, when a new release of your Logi product comes out, is to select only the Server Engine.

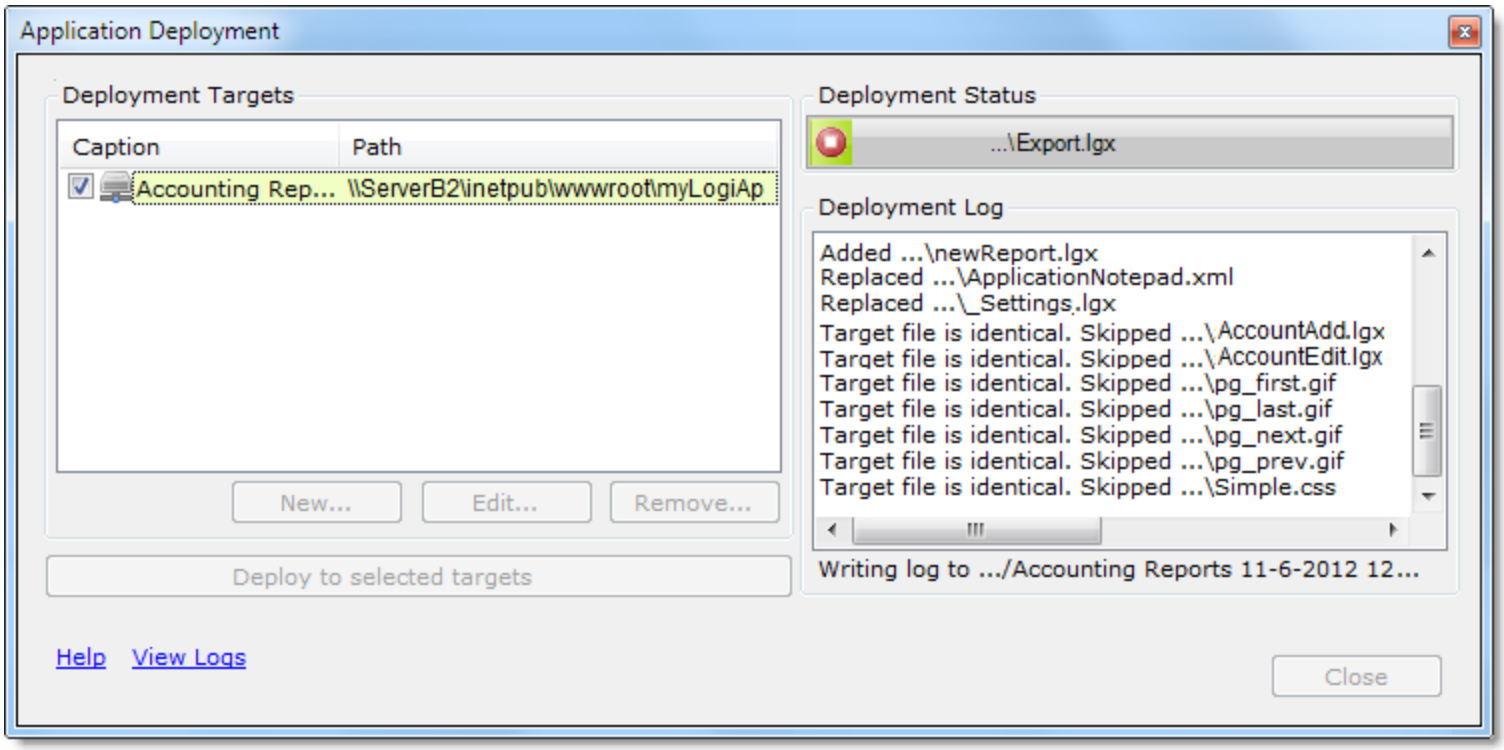
If the destination requires a user ID and password in order to access it, when you click the Test button, you'll be prompted for login credentials. If the credentials are successfully authenticated, the user ID will automatically be saved with the deployment target details and, optionally, the password as well.

Once a deployment target has been defined, it appears in a list:

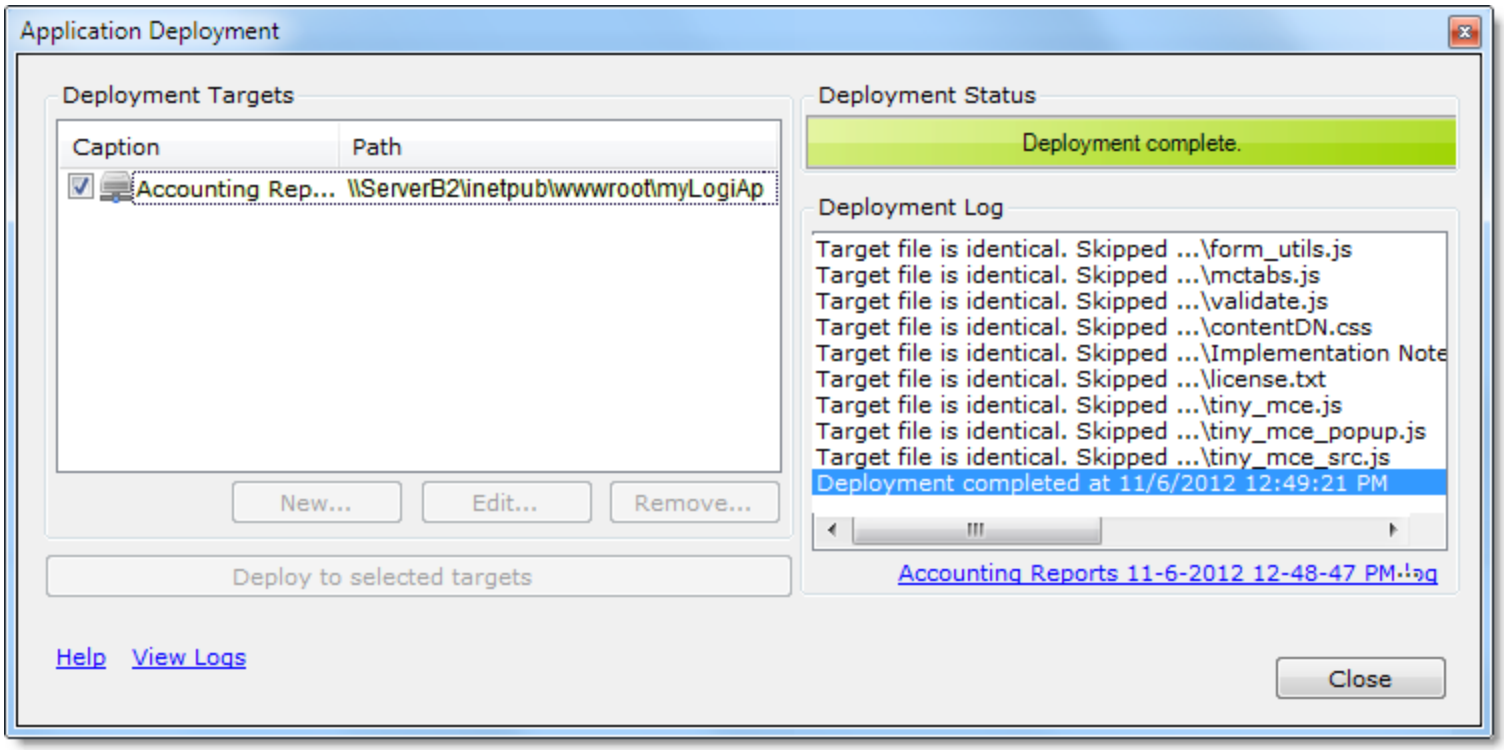


Once a deployment target has been defined, it appears in a list, as shown above, where they can be managed and edited. To run a deployment, check its check box and click the **Deploy** button. If multiple deployments are selected, they'll run consecutively.

The connection with the destination server will be established and then the files will be deployed to it.



The Deployment dialog box will expand to display the results of the process, as shown above. Text log files, stored in the rdDeployment folder beneath each Logi application folder, are accessible through the **View Logs** link, and document each deployment.



Finally, the deployment will be complete, and the link to the log file will be displayed.

It's very important that you understand that copying the files, using the Deployment Tool, *does not register the application* with a web server on the target machine. So, after the first deployment, this has to be done by the developer using Studio, Server Manager, or the web server's tools.

Logi Java Application Folder Permissions

After deployment of a Logi Java application, you can elect to grant 775 (full access) permissions manually on the application root folder and all of its child folders and files. If you prefer a more selective approach, typical Logi application folders require these permissions:

Folder	Permission
Logi app root	440
assemblies	
rdDataCache	660
rdDownload	660
rdTemplate	660
WEB-INF	
lib	
PhantomJS (binary)	550
_Definitions	
_SupportFiles	

Folder	Permission
goBookmarks (InfoGo app only)	660
(your custom folders)	660

Controlling Which Files Are Deployed

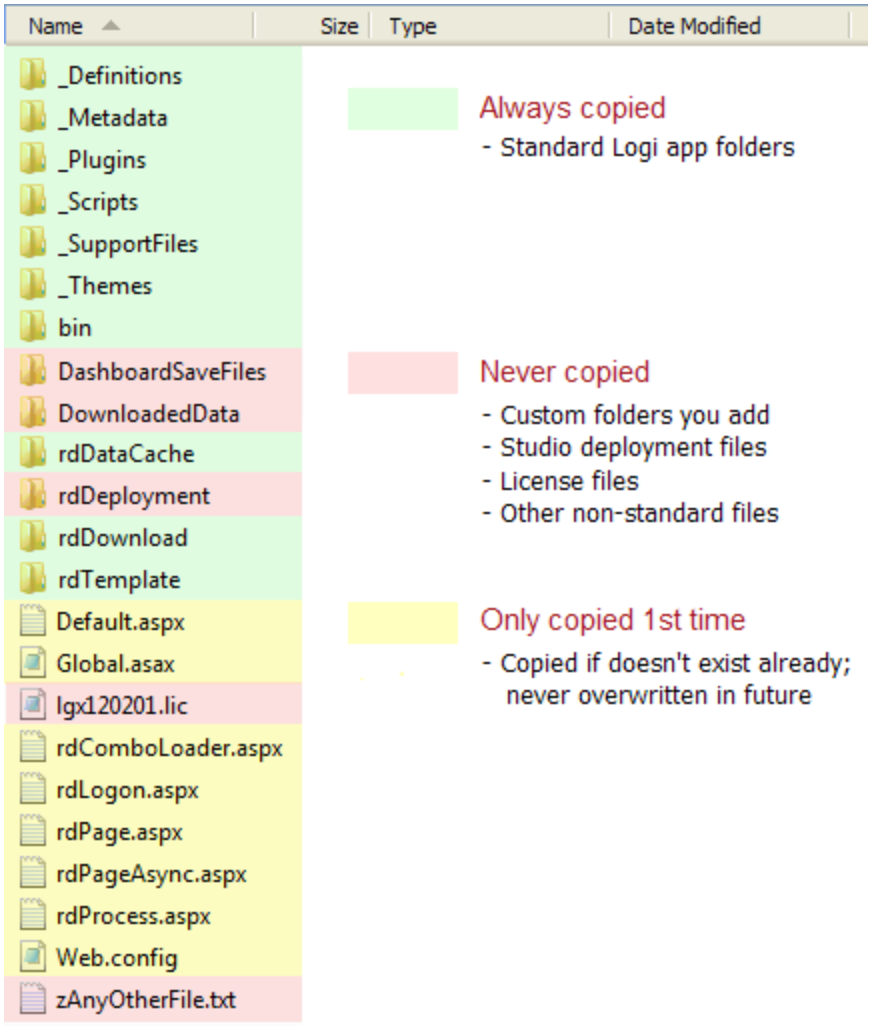
The Add/Edit Target dialog box allows you to select the files to be deployed by category:

Category	Description
Application Definition and Support	Copies all files in all folders with names beginning with an underscore, such as _Definitions, and _Support Files. Identical files will be skipped.
Application Settings	Copies _Settings.lgx only. Identical files will be skipped.
Server Engine (.NET app)	Copies the rdTemplate and bin folders - identical files will be skipped. Also copies any .aspx, .asax, and .config files if they do not already exist; does not overwrite them if they do exist.
Server Engine (Java app)	Copies rdTemplate, Assemblies, and WEB-INF folders - identical files will be skipped. Also copies any .aspx, .asax, and .config files if they do not already exist; does not overwrite them if they do exist.

Why not deploy *all* of the categories *all* of the time? You'll want to do so for your first deployment but, generally, you won't want to for subsequent deployments in order to avoid overwriting files that may have been *customized* for the production server, such

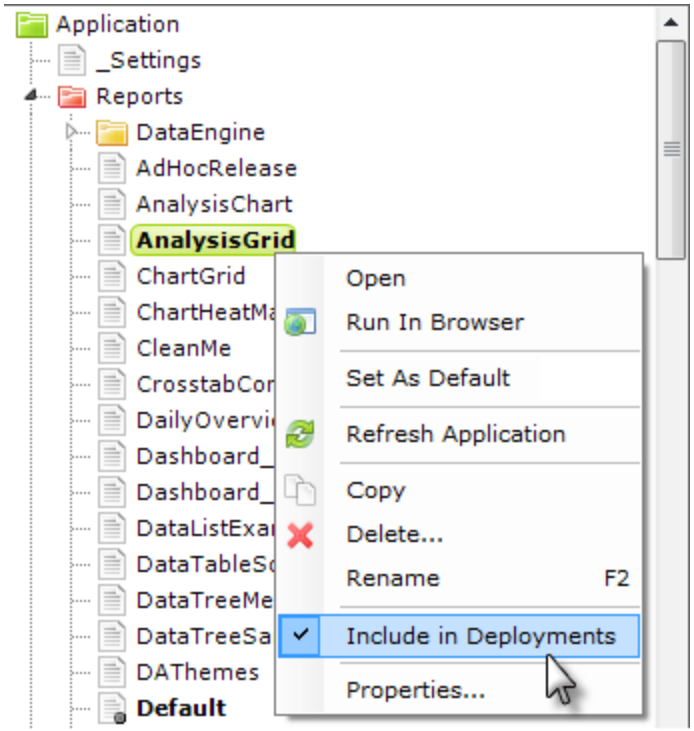
as _Settings.lgx. And, unless you're changing engine versions, deploying the engine files with every deployment is unnecessary and a waste of time. So the tool offers you the opportunity to tailor your deployments as needed.

The Deployment tool will *not* copy *every* file in your application folder:



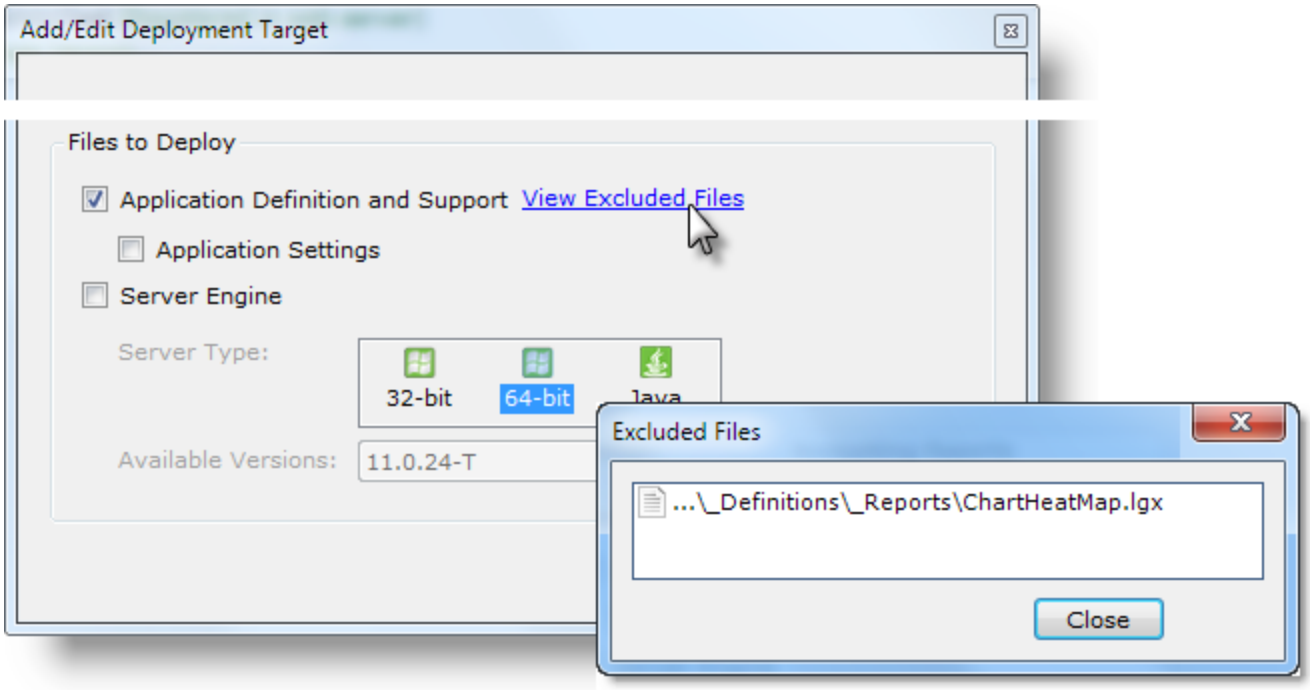
The image above illustrates which folders and files are copied. Files are compared during the process and identical files will be skipped. Of special note is the fact that custom folders you may add to the application, for example to hold Dashboard Save files, *are not copied* and you will need to create these during your first deployment manually. Other files you may create in the application folder that are not stored in folders will *not* be copied, as well.

However, developers may want to *exclude* specific application definition and support files from a deployment, and Studio provides an easy way to do that:



By default, all files in Studio's Application Panel are included in deployments. However, if you wish to *exclude* a file, select it in the Application Panel, right-click it, and click the **Include in Deployments** item in the popup menu, as shown above.

You can also exclude an entire *folder* in the Application Panel in the same way. Suppose you want to deploy just 1 or 2 report definitions. This can be done by first selecting the _Reports folder, right-clicking it, and clicking the checked Include in Deployments item, which unselects *all* report definitions. Then select and right-click the 1 or 2 report definitions you want to deploy, and click their Include... items to include them in the deployment. Then run the Deployment Wizard.



If files have been excluded, the **View Excluded Files** link, shown above, will be displayed to alert you to this situation when you add or edit a deployment target in the Deployment tool. Clicking the link will display a list of the excluded files.

License Files

License files are *never included* in deployments, so you will have to manually deploy your license file the first time you deploy your application, or configure it to use a centralized license file. For more information about license files, see *Product Licensing*.

After a First-Time Deployment

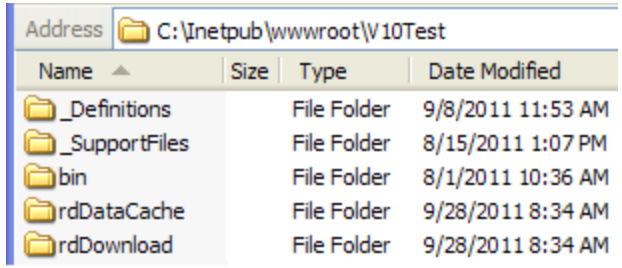
After an application is deployed for the first time, you must make some configuration changes to ensure it will run. See "Post-Deployment Server Configuration" on page 131 for details.

For subsequent deployments, you need only run the Deployment tool to update the files on the target server.

Alternate Method: Manually Copying Your Info Application

If you don't have a version of Studio that includes the Application Deployment Tool, you can use another deployment approach that works well if you can map a shared network drive on the server: just *copy* your entire Logi application folder to the production server.

From an operational perspective, for .NET applications, it doesn't really matter where you put the folder on the server but other company-specific considerations (storage location conventions, security, space limits, etc.) may apply. For Java applications, depending on your specific web server, you may be required to place your app folder in a specific folder (for example, beneath the `webapps` folder for Apache/Tomcat).



Just be sure that your copy operation gets *all* of the files and folders inside your application folder to the production server.

If you develop your application on a 32-bit machine and copy it to a 64-bit production machine, *do not* copy the entire application folder, as some Logi Server Engine files in your app will be 32-bit-specific. Instead, either use the Deployment Tool and ensure that you deploy 64-bit engine files, or install Studio on your production server, use the New Application Wizard to create an empty 64-bit application, then copy only your definition and support files over from your development machine.

If you're distributing your application to a remote site, you can use a compression utility like **WinZip** to package your entire Logi application project folder into a single .ZIP file.

This approach copies *all* of the Logi Server Engine files as well as the application files, so the version of the production application will be the same as that of the development application. This approach will also copy any license file in the application root folder to the production server, where it may not be viable.

Naturally, if you don't have a network connection to the production server, you can also use other methods, such as burning the entire application folder to a DVD or copying it to a USB "thumb drive" in order to move it to the production server.

Deploying a Logi Services Database

If your application uses **Logi Services** technology (i.e. includes the Discovery 3.0+ add-on module), you'll need to deploy a copy of the Platform Database (PDB) to your production server. The PDB contains a variety of objects, including security information, Dataviews, settings, licenses, and more.

 These deployment instructions *copy* the development PDB; they do not make *incremental updates* to the production PDB. Use caution if you're considering repeating these instructions at a later date; you want to avoid overwriting new records in the production PDB with "old" data from the development PDB.

Use these following steps to deploy the PDB. They assume a default installation location on both servers.

1. Stop the Logi Application Service and Logi Data Service services on both servers.
2. On the production server, create a backup folder and *move* the following folder (with all files and sub-folders) and files into it:

   ```
   C:\LogiAnalytics\Discovery\platform\ldap
   ```

   ```
   C:\LogiAnalytics\Discovery\platform\db\LogiDB.mv.db
   C:\LogiAnalytics\Discovery\platform\bin\logiconfig.bat
   C:\LogiAnalytics\Discovery\platform\settings\logiDataService.json
   ```
3. Copy the `C:\LogiAnalytics\Discovery\platform\ldap` folder (with all files and sub-folders) from the development server to the same location on the production server.
4. Copy these files from the development server to the same location on the production server:

   ```
   C:\LogiAnalytics\Discovery\platform\db\LogiDB.mv.db
   C:\LogiAnalytics\Discovery\platform\bin\logiconfig.bat
   C:\LogiAnalytics\Discovery\platform\settings\logiDataService.json
   ```

5. Ensure that the copied folder and files have the same file access permissions as neighboring folders and files (or are unblocked) on the production server.
6. Restart the Logi Application Service and Logi Data Service services on both servers.
7. On the production server, test to see that the Admin user can access Dataviews stored in the PDB.
8. Delete the folder and files in the backup folder from Step 2 once you're sure the application is working correctly.

 The PDB contains the product license, which is machine-specific. Once you copy it to another machine, you will need to replace the license in the PDB (assigned to the development server) with one assigned to the product server. See *Product Licensing* for details.

There is a method of migrating the PDB to a production server *without* the security information. Alternate security information can then be inserted from an Excel spreadsheet. This is done using a script which is described in the platform documentation, in a topic entitled *Migrating the Platform Database*. To access this documentation from Logi Studio, run the Dataview Authoring tool, log into it, and click the Help icon in the upper right-hand corner near your user name.

Deploying to Cloud-based Platforms

Windows Azure and Amazon Web Services (AWS) are examples of a "cloud-based services platform". They allow you to run applications from, and store data on, cloud servers. It's possible to deploy your Logi Info application to some of these services and run it from the cloud.

Logi Info apps will not run on Azure Platform-as-a-Service offerings. Logi does not specifically offer standard support for this kind of deployment; however, you may care to engage Logi Professional Services to assist you.

For more information, see *Cloud Deployments*. If you want to bundle the Logi Analytics Platform and Logi application to execute from a container environment, see *Container Deployments*.

Post-Deployment Server Configuration

Once you have physically deployed your Info application to the production server for the first time, you need to configure it to run in the new environment.

When Using Windows IIS Web Server

1. Provide a Logi Info license file for the application, unless you're using a centralized v12 license file.
2. *Register* the application with IIS. You can do this using Studio (if you've installed it on the production server), Server Manager, or the IIS Manager tool.
3. Ensure that the datasource connection strings or parameters, constants, and other server-specific attributes are correct in your application's _Settings definition. This can be done using **Notepad** if Studio is not installed.
4. If your application writes any output to the file system outside of your Logi application folder, ensure that the account the web server uses to run your application (ASP.NET, NETWORK SERVICE, or Application Pool, depending on IIS version) is granted *Write* permission to the relevant folders.

When Using a Java Server

1. Provide a Logi Info license file for the application, unless you're using a centralized v12 license file.
2. Provide any additional configurations required for your server. See *Java Server Configurations* for specific information.
3. Ensure that the datasource connection strings or parameters, constants, and other server-specific attributes are correct in your application's _Settings definition. This can be done using any text editor.
4. If your application writes any output to the file system, ensure that the identity the application runs under has appropriate file access permissions for the relevant directories.

Subsequent deployments for the purpose of updating report definitions, images, XML data, etc. should not require you to do any of the steps discussed above, unless you're changing folder locations or _Settings attributes. This means Logi Server Engine files and _Settings.lgx can be left out of future deployments undertaken for simple update purposes.

The Logi Server Engine

The **Logi Server Engine** is a file set, included with each Logi application, that extends the functionality of the web server. The files are processed at runtime and are usually cached by the web server for best performance. As web server extensions, they provide the server with the ability to process Logi definitions and data to generate an appropriate output. The **Logi Data Engine** is the part of the Server Engine that handles data retrieval, temporary caching of that data as XML, and manipulation of the data such as filtering and grouping. This topic discusses important aspects of the Server and Data Engines.

The following topics discuss additional features of the Server and Data Engines:

- Hybrid Caching
- Parallelization

About the Server Engine

The Server Engine is a collection of files that are part of every Logi application. As a result, multiple applications with different versions can co-exist on the same web server, and applications can be independently up- or down-graded to different versions. This has proven to be a worthwhile trade off for the additional disk space consumed by (possibly) redundant engine files.

The Data Engine is embedded within the Logi Server Engine and doesn't appear as a separate component during product installation. However, developers who have a good understanding of how it works may be able to take advantage of certain features in order to improve the performance of their applications.

At runtime, the Data Engine handles making a connection to data sources and retrieving the raw data. The raw data may be cached in memory or on disk depending on the *"Hybrid Caching" on page 135* scheme. The cached data is then subjected to filtering, grouping, and other required manipulations before its ready to be rendered by other portions of the Server Engine.

The Debug Trace page displays distinct milestones for each phase of the data retrieval and manipulation process, and includes links that the developer can use to view the actual data in each phase. This is very useful in understanding the effects of and, given that a time scale is presented, the processing time required for, each operation.

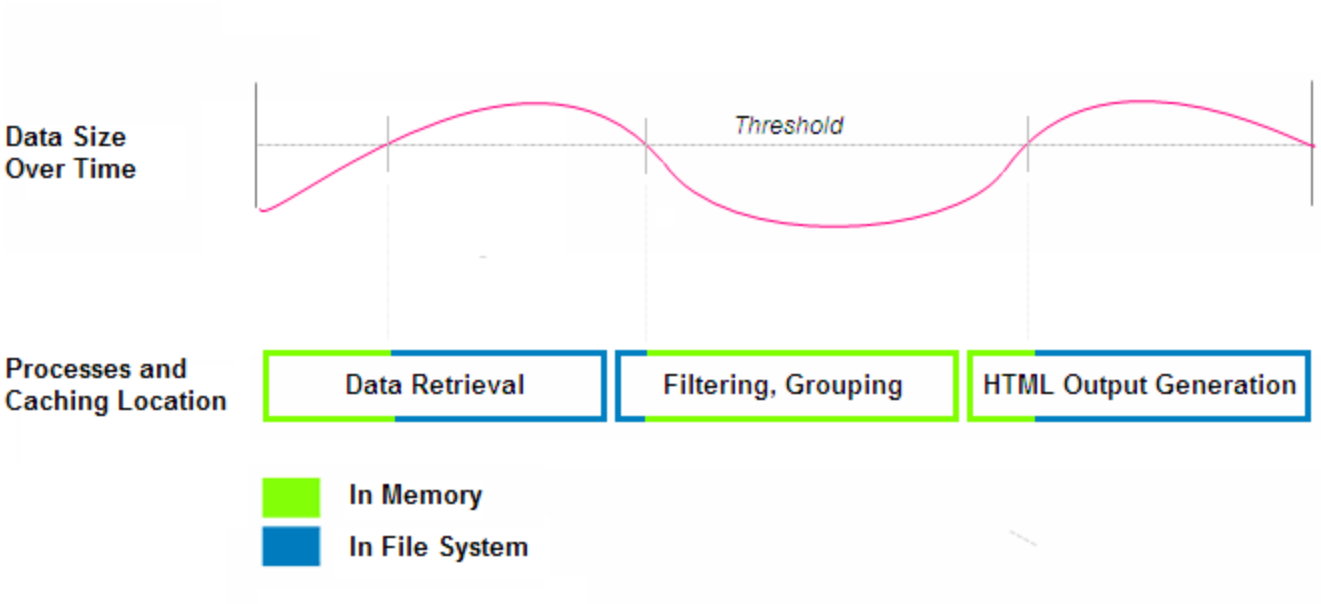
The Server Engine includes a companion utility, **Server Manager**, which provides all the necessary application management features typically necessary on a production web server. This makes it unnecessary to license and install Logi Studio on the production server.

Hybrid Caching

The Logi Info Data Engine includes our "hybrid caching" feature, which provides improved management of web server resources, especially memory. When the a Logi application is responding to a request for a report, there are three distinct production phases that involve data manipulation:

1. When a datalayer *retrieves* data and *caches* it as an XML stream,
2. When that cache is *processed* using filtering, grouping, and aggregation,
3. When the HTML output (the report page) is being *generated*.

These activities typically use web server **memory** but, if the amount of data involved is very large, the potential exists to literally run out of memory. The Data Engine manages this situation as illustrated below:



The hybrid caching feature *prevents*out-of-memory conditions by allowing the data streams used in these operations to be dynamically switched between the web server's **memory** and the web server's **file system**. The engine continuously monitors the amount of data it's manipulating in each production phase and, if it exceeds a specific threshold, switches caching to the file system. If the amount of data drops below the threshold during subsequent phases, caching switches back to memory. Logi report developers can use the default memory consumption threshold (10MB) or set **special constants** in their application to tailor the threshold as needed.

Configuring Hybrid Caching

The following constant can be added to the **Constants** element in your _Settings definition to configure the hybrid caching of data and rendered HTML:

Name	Description
rdMemoryStreamLimit	This value sets the threshold, in MB, for shifting from memory caching to file system caching. A value of *0* will force all data and HTML caching to the file system; a value of *2048*, the maximum, will force it to memory. The default value is *10MB*.

In general, the default setting is good for most purposes. Changes should be considered *only* if actual problems with memory or performance are observed. Switching to 100% memory-based caching may sound appealing from a performance perspective but may not actually provide the best overall use of resources in your server.

 The rdDataCache and rdDownload system folders are used to store a variety of other temporary files. Changing to 100% memory-based caching of data and HTML will *not* prevent these other files from being written to these folders.

Parallelization

Logi v12 includes advances in "parallelization", the ability to execute multiple actions simultaneously. This translates into improved performance for the user. This technology improvement has been applied to the way the Logi Engine handles datalayers and Dashboard and tab panels.

Datalayers

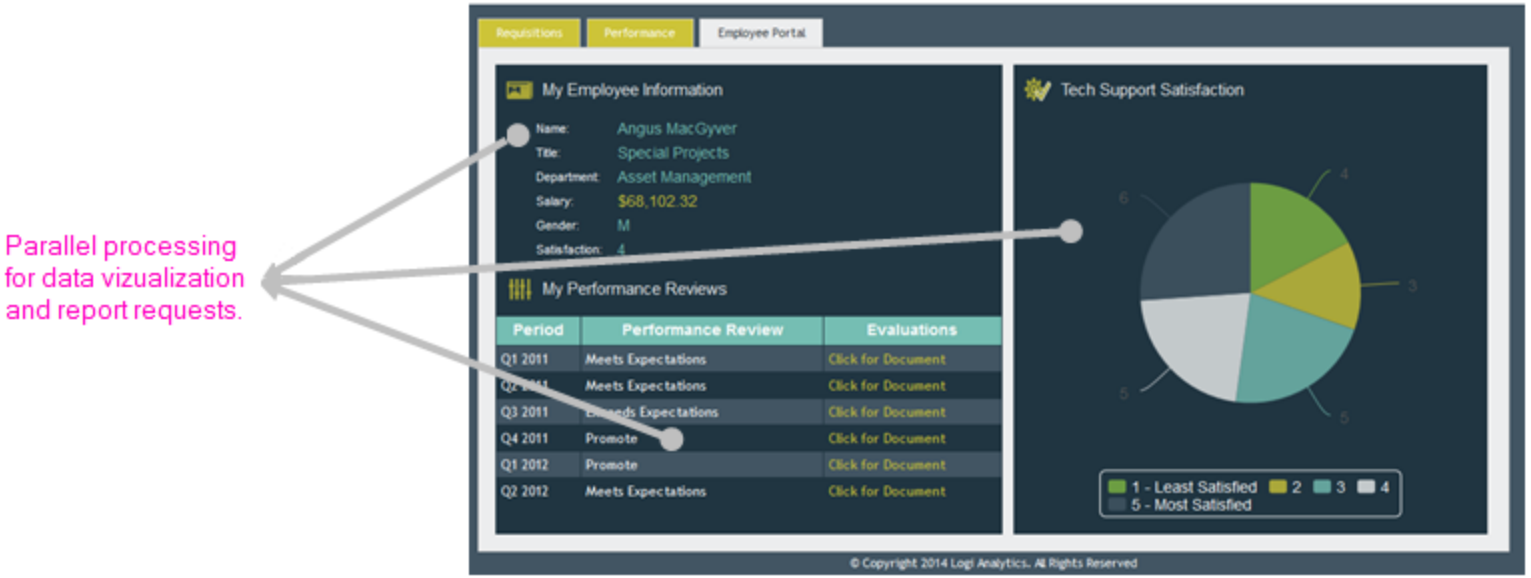
Data retrieval through datalayer elements is bounded by the longest performing query, rather than the *number* of queries. Multiple threads are launched at the web server to satisfy multiple requests and execute in parallel.

Load XSL	Success	True	
Get XML DataLayers			
	Launch DataLayer Threads		
	Thread for DataLayer ID: firstDataLayer	View Debugger Trace	.071
	Thread for DataLayer ID: secondDataLayer	View Debugger Trace	.079
	Thread for DataLayer ID: thirdDataLayer	View Debugger Trace	.089
	Threads Completed		
Finalize DataLayers			

These threads, and their execution times, are shown in the standard Debug Trace page, as shown above. This improves performance in pages with input elements, Data Tables, and data lists.

Dashboard and Tab Panels

Dashboard and tabbed panel loading benefits greatly from parallelization.



Dedicated threads for each panel allow parallel processing of panels, which improves performance, particularly when each panel has local parameters. Testing of complex reports and Dashboards has shown up to a 4x performance improvement when upgraded to v12 to take advantage of Logi parallelization technology.

Product Life Cycle Expiration

All Logi Analytics products have a life cycle. The life cycle begins when a product is released and ends when Logi Analytics no longer provides maintenance releases, updates, or technical support. This topic describes our life cycle policy, and what you can do if your product is no longer supported.

Logi Support accepts Support Cases for all versions of the last two *major* releases (for example, all Logi Info v23 and v14 releases are currently supported, but v12 and earlier are not).

If you are using products or versions whose life cycle has ended, you should discuss your needs with the Logi Sales Team. If you have questions about your software version, contact CustomerService@LogiAnalytics.com.

Life cycle management allows us to focus on current and future versions of our products, to manage our resources to best support our latest and most popular product versions, and to ensure customer satisfaction with our Support staff by seeing that they're well-trained on our latest products and technologies.

Per the Logi Analytics Support Policy, Logi Info v11 has reached End of Life and support for this product will be phased out with the release of Logi Info 14. This End of Life process concludes on 12/31/2021. Please contact Sales or Customer Service for information about upgrades. Thank you.

Support Coverage

If you are one of our Support Plan customers using an unsupported release, we will assist you. But, if the root cause of your issue is a bug in our product, we will advise you to upgrade to a supported release. Service Packs are released for 18 months after the initial product release, unless separately contracted differently. For more information about current product release levels, see *Current Releases*.

32-bit Products No Longer Available

Hardware manufacturers have started focusing almost exclusively on 64-bit server platform architectures and major OS manufacturers have transitioned their products to 64-bit only versions, making it difficult for us to comprehensively test 32-bit versions of our products against them. As a result, with the advent of Logi Info v12.2, we no longer provide 32-bit versions of our software.

Frequently Asked Questions

The following topics provide answers to frequently-asked and entry-level technical support questions:

- System Requirements, Installation, and Licensing
- Application Configuration and Settings
- Working with Data
- Report Design
- Working with Support Files(Style sheets, Images, XML, etc.)
- Other Questions

System Requirements, Installation, and Licensing

This topic provides answers to frequently-asked questions about system requirements, installation, and licensing:

1. What Operating System versions are supported?

Supported OS versions include:

- Windows Server 2019 (v12.7 + only), 2016 (v12+ only), 2012 R2, 2008, 2003
- Windows 10 (with Logi Info v12+ only)
- (all editions except *RT*)
- Windows 7-8 (all editions)
- SUSE, Red Hat, Ubuntu, CentOS, and most other flavors of Linux

2. Are there 64-bit versions of your products?

Yes. Starting with v12.2, only 64-bit versions are available. Hardware manufacturers have started focusing almost exclusively on 64-bit server platform architectures and major OS manufacturers have transitioned their products to 64-bit only versions, making it difficult for us to comprehensively test 32-bit versions of our products against them. A 64-bit environment and software offer numerous benefits relating to improved performance.

3. What are the licensing ramifications of using server virtualization?

Please see *Server Virtualization*.

4. Will a Logi application run in a shared hosting environment?

Logi applications require a High Trust Level to run. Most hosting centers insist on Low or Medium Trust levels on shared web servers for security purposes. However, if the hosting center offers a dedicated server or virtualized server, they may allow a High Trust Level and Logi applications can then be used.

5. Do you have a version of your products that will run on Linux with the Apache-Tomcat web server?

Yes, Logi Info can create web applications that use OracleJDK or OpenJDK 8, 11, 12, 13, or 14 instead of .NET and run under Linux or other UNIX derivatives, on Apache-Tomcat and other open source web servers. Fore more information, see *About Logi Apps and Java* and *Java Usage Policy*.

6. When I register my application, its virtual directory is created on IIS under an "OWA" web site. Why is this?

By default, IIS is installed with a "Default Web Site" configured, which has an ID of #1. The Logi tools register applications (create their virtual directories) under the web site with ID #1. If your Default Web Site is missing and virtual directories are being created under "OWA" (which stands for Outlook Web Access and is used for remote email access) your IIS configuration has been highly customized. You need to get a trained IIS administrator involved in order to configure this server for use with Logi apps.

7. Should I install Logi Server on the same web server I use for Outlook Web Access (OWA)?

No. Microsoft recommends that, for reasons of security, performance, and scalability, you do not run other web sites or web applications on your OWA web server platform.

8. How are your products delivered after someone purchases them?

Product delivery is via download from our web site; documentation is available on this web site. We do not distribute our products on CDs or DVDs.

9. Will your products work via a Citrix server?

Yes. Logi applications, as web applications, can be served and browsed in a Citrix environment without any problem. Our Studio development tool, as a standard Windows client application, needs to be installed and published as a Citrix application. As is the case with many apps published this way, its performance will be a little less snappy than it would be if it was installed and run from a user's local machine.

10. Will your products work with Microsoft SQL Server?

Yes. Our products have been tested with and work with all versions of MS SQL Server, starting with SQL Server 2000.

11. I received this error message: Thread was being aborted

Depending on the availability of system resources, the IIS worker process may not have been able to acquire enough memory to process the request and recycled. If your web server is IIS 6, you may want to investigate how its Application Pools are being used. If you have a large number of web applications all using the Default Application Pool, you may want to isolate some, or all of them, in separate app pools.

12. It appears I need a copy of the Logi license file in *every* Logi app folder. Is a centralized file possible instead?

Yes, instead of distributing and maintaining many license files on a single computer, you place a one copy of the license file in a central location. In each Logi application, you modify the _Settings definition, configuring the General element's License File Location attribute to point the folder containing the license file. The account the web server uses to run your apps must have access permissions to that license file folder.

Application Configuration and Settings

This topic provides answers to frequently-asked questions about application configuration and settings:

1. In Studio, the Application page shows a "Warning: Version Mismatch" message. What does this mean?

This warning means that the application was created with a version of the Logi product that is older than the version of Studio currently installed on your computer. As a result, some application features available in the newer product may not be available. To upgrade your application to a different Logi product version, click the Change Version... link on Studio's Application page, or use the Server Manager utility, available on the Studio Tools menu. For more information, see *Logi Product Upgrades*.

2. How do I turn on debugging?

You can turn on debugging in Logi Studio for all your report definitions by clicking the **Debug** icon on the menu or toolbar and selecting *DebuggerLinks* from the selection list. Clicking the icon sets an attribute value in the _Settings definition, General element, and you can also turn debugging on by setting that value directly.

3. How can I switch between JavaScript and VBScript?

Support for VBScript has been deprecated - it continues to be supported but JavaScript is the recommended scripting language. JavaScript is the default scripting language. The choice of scripting language is set in the _Settings definition, using the **General** element's **Scripting Language** attribute. When creating a Logi app for Java, JavaScript is the only scripting language choice.

4. Is there some way to assign a style sheet to an entire Logi application?

Yes. If you prefer to do that, rather than assign style sheets in each individual report definition, you can do so in the _Settings definition. Just add a **Global Style** element and assign the style sheet file to it, and it will then apply to every report definition.

5. I'd like to create a mailto: link in several reports. Is there a way to define the email address as a constant?

Yes. Constants can be defined in the _Settings definition and referred to from report definitions. This allows the value of the constant, in your case the email address, to change, if necessary, without requiring a change in every report definition. Constants are referred to using the @Constant token.

6. What happens when you "register this application" using the Studio wizard?

When using IIS as your web server, the registration wizard creates an IIS Virtual Directory for your web application. You must be logged-in with an account that has administrator privileges in order for the wizard to create and configure IIS virtual directories. You can also do this manually, if you prefer, using the IIS Management tool provided with Windows.

7. How can I browse my report from within Studio? Whenever I try to do it, the Default report is displayed.

In the _Settings definition, the **Application** element has an attribute that selects which report is considered the "default" report and is displayed when you select Browse from Studio's toolbar. You can quickly change this selection by selecting and right-clicking a report definition in Studio's Application panel, then selecting "Set As Default" from the pop-up menu that appears. You can also browse any individual report definition by selecting and right-clicking it and selecting "Run in Browser".

8. I noticed the Team Development element in _Settings. What does that do?

Team Development is a simple source code control system. It was designed to be used in an environment where there are multiple developers working on the same application and it's useful to control access to definitions and support files. When Team Development is enabled, application files are locked and developers must check them in and out to work on them. This prevents overlapping editing of the same file and creates an audit trail of who worked on which files. File History can also be enabled so that revisions can be rolled-back and undone. More information is available in *Using Logi 12 Studio*.

9. Is it possible to use Logi report definitions that are stored in a database table?

Yes. Logi report definitions are XML files and these can be stored in a table column. The **External Definitions** element can be used in the _Settings definition with a datalayer element to retrieve the stored report definitions and make them available to the application.

10. When I ran my report I received an "Access denied" error message.

In order to allow the account used by the web server to run Logi applications to write temporary cache files to the server's hard drive, you must give it permissions to do so. This is described in detail in *File Access Permissions*.

11. Can I configure my Logi application's document type declaration (doctype)?

Yes, the _Settings definition's **General** element includes a **Doctype Declaration** attribute, that can be set to *Html5*, *Xhtml Strict*, *Xhtml Transitional*, *Xhtml Frameset*, or *None*. If left blank, the default is *Html5*.

Working with Data

This topic provides answers to frequently-asked questions about working with data:

1. In Studio, when adding Data Table rows, do I have to add a datalayer for each row ?

No. A datalayer, for example, that issues a SQL query will contain all of the rows and columns that were returned in the result set. Your report definition needs to contain Data Table column and label elements for each column you wish to display. When the report is browsed, the server will iterate through and display all rows in the datalayer automatically. See *Data Table Tutorial - Manual* for an example of how to create a definition that does this.

2. Can I run a Stored Procedure?

The DataLayer.SQL element only seems to accept SQL statements.
In general, Stored Procedures should be run using the DataLayer.SP or Procedure.SP elements. Input and output parameters can

then be used with stored procedures using the SP Parameters element.

3. My SQL query is returning DateTime values in this format "2008-08-07T14:20Z". How can I work with this?

DateTime-type data returned into a data layer by a query against a SQL data source is usually in ISO 8601 format. If you wish to use script functions to compare, manipulate, or format the date data, you need to convert it into a compatible format. Logi's intrinsic **CXMLDate** function has been provided for this purpose. See *Special Functions and Attributes*. We also offer the *Time Period Columns*element, which makes it easy to parse data, such as hours or months, from this format.

4. Do Logi products support the MS SQL Server 2005 XML data type? Can you search within the XML data?

Yes. Our products support *any* valid SQL statement, so queries that utilize XQuery functions and XPath expressions, used to search in and work with XML data in XML data type columns, are supported.

5. What formats can you export reports and/or data to?

Logi Info can export to .pdf, .csv, .xml, .xls/xlsx, and .doc/.docx files and can insert data into predefined PDF, Word, and Excel templates.

6. Do Logi products work with BLOBs stored in table columns?

Yes. The File Column element allows BLOBs and CLOBs stored in table columns to be retrieved and used. For more information, see The File Column (BLOBs).

7. Do Logi products support the MEDIAN function?

Once data has been retrieved from a datasource it can be manipulated using a the Aggregate Column element, which supports Average, Concat, Count, Distinct Count, Maximum, Median, Minimum, Mode, Standard Deviation, and Sum functions. Aggregations can include or exclude columns with Null values. By default, they're *excluded*. Create the constant `rdCalculationsIncludeNulls`and set it to *True* if you want to include them.

8. How can I parse the parts of date and time data values?

Once data has been retrieved from a datasource it can be parsed using the **Time Period Column** element. This element will add a column to the datalayer that can contain parts of a DateTime value, such as: *Year*, *Quarter*, *Month*, *MonthAbbreviation*, *MonthName*, *DayOfYear*, *DayOfMonth*, *DayOfWeek*, *DayOfWeekAbbreviation*, *DayOfWeekName*, *Hour*, *Minute*, *Second*, and many more. For more information, see *Time Period Columns*.

9. Is there some way to get a list of the report definitions within a Logi application, as data?

Yes, the DataLayer.Definition List element will produce, as rows and columns, information about the properties of each report, process, widget, and template definition in the application. This data can then be used like any other data, for example, in reports, menus, and selection lists. For more information, see *DataLayer.Definition List*.

10. How can I create simple options for use with Input Select Lists? I don't want to build a table for three options.

There are two easy ways to do this. First, you can use a DataLayer.Static element beneath your Input Select List, then add one child Static Data Row element for each list option. Second, a slightly more flexible approach is to create a simple XML data file that contains your options and then use a DataLayer.XML File element beneath your Input Select List. This has the advantage that the options can be easily edited or expanded with having to run Studio and change the application.

11. My table contains dates entered as text, but I want to be able to sort by date.

How can I do this? Retrieve your data into a datalayer element, then add a Calculated Column element beneath it. In the Calculated Column element's Formula attribute use the Logi intrinsicCDATE( ) function to add a column to the datalayer that contains your text dates converted to proper DateTime data. The calculated column can appear in the Data Table in your report with a header link for sorting by date.

Report Design

This topic provides answers to frequently-asked questions about report design:

1. Why would I use Layout Positioning instead of Flow Positioning?

Layout Positioning allows you to place elements in absolute positions on the screen. However, the default Flow Positioning, which bases each element's location on that of previous elements, produces fewer browser and device compatibility issues. We don't generally recommend use of Layout positioning. *Important Note:* If you convert a report definition from Flow Positioning to Layout Positioning and then switch back, your report may not be perfectly converted back to its original state. This is due to certain attributes that Layout Positioning inserts, which become ambiguous in the reverse conversion.

2. I generated my first report in Studio using the New Application Wizard. Now how do I add a second report?

One Logi **application** can contain many **report definitions**. The New Application Wizard created both your Logi application and your first report definition. To add additional report definitions to your application, on the main menu click New File → Report.

3. Can end users create, generate, design, format and customize reports by themselves?

Yes, using two methods. First, if you develop a report using Logi Info and include a "super element", such as the *The Analysis Grid for End Users*, end users can significantly customize the report at runtime. Second, we also offer add-on modules, extensions of Logi Info that provide self-service reporting, allowing users to create their own analyses without any developer assistance. For more information on the add-on modules, see "Add-on Modules" on page 69.

4. Can Logi reporting products open/use Crystal Reports .rpt files?

No. Crystal Reports .rpt files are proprietary, binary files and completely unlike the simple XML text files used by Logi reporting products.

5. Can I include data entry fields in my report? Can I "write back" (update) my database using them?

Logi Info supports HTML forms and standard user input controls, including text boxes, text areas, check boxes, radio buttons, and selection lists. Data entered or selected by the user can be used to dynamically change reporting criteria and can also be used to insert new records or update existing records in database tables. Logi Info includes features (Process tasks) that make this very easy.

6. Is there some way to include common report parts, like headers and footers, from a single source?

Yes. Logi products include a feature called **"**Shared Elements" which allows you to establish a set of elements, such as a report header with logo and date, as "shared". These can then referenced in all report definitions in an application. Any subsequent changes made to the single instance of the shared elements will then ripple through to appear in all reports that use them. For more information, see *Shared Elements*.

7. How can I hide certain sections of reports based on dynamic criteria, such as data values or a user profile?

One approach is to use the Division element, which acts as a "parent" container for parts of reports, and controls their visibility. For more information, see *Work with Conditions*.

8. Some of my chart Labels are very long, requiring a big chart bottom border I don't want. What can I do?

Our charts include a Max Label Length attribute that specifies the maximum number of characters that will be displayed for a label before the text is trimmed and the remainder replaced with an ellipsis (...).

9. Can I have multiple Analysis Grids in my report definition?

No, you can only have one "super-element", such as an Analysis Grid, per report definition. This is because of internal identifiers within the super-elements, which would cause confusion if multiple elements were allowed. You can, however, place Analysis Grids on several report definitions and then include them, as SubReports in *IncludeFrame* mode, in a single master definition.

10. Does a user have to login to Logi Ad Hoc to use and create reports?

Yes, Logi Ad Hoc is a web application and users do generally need to login to it in order to use it. Ad Hoc security is fully configurable, however, from none to fully-integrated with external security schemes.

11. Is there a way in Logi Info to format numbers using metricnotation, such as 10K?

Yes. The options available in Format attributes include "mp", which stands for Metric Prefix and which will automatically truncate numeric values and append appropriate characters, such as "K" or "M". For more information, see *Format Data*.

12. It would be great if a chart legend could be clicked to show/hide different data series. Is that possible?

Yes. Logi Info v12+ includes "legend filtering", which allows the user, at runtime, to click entries in the legend, toggling the inclusion of data in the chart, which is immediately redrawn. The feature is enabled using the Legend element's Legend Filter attribute, and applies to static Pie, Line, Area, Bar, Scatter, and Polar charts.

Working with Support Files

This topic provides answers to frequently-asked questions about working with support files:

1. I tried to add an Excel template to my application and received an "Access denied" error message.

This can occur if you still have the template file open in Excel while trying to add it to your application's Support Files. Close the file in Excel before trying to add it to your project.

2. I tried to save data in an XML data file and received an "Access denied" error message.

In order to allow the account used by the web server to run Logi applications to write to the server's hard drive, you must give it permission to do so. This is described in detail in *File Access Permissions*.

3. Where did the files I saved in rdDownload go? They've been deleted.

Two folders, rdDataCache and rdDownload, in your Logi application folder are used by the server engine to store *temporary* files. Any files in these folders are automatically "cleaned up" (deleted) periodically. If you want to write data out to files or export reports and retain them, we recommend you create a new folder in the application folder and place them there.

4. What kinds of image files can I use with Logi applications?

Logi apps will work with image files in these formats: .gif, .jpg, .jpeg, .png, .bmp, .wmf, .xbm, .art

5. Can #ID type class definitions be used in style sheets with Logi applications?

Yes. For complete information about the class definitions that can be used with Logi apps, see *Using Style Sheets*.

6. Can external HTML files be embedded within Logi reports?

Yes. The **Include HTML** and **HTML File** elements can be used to embed HTML code and external HTML files, respectively, into Logi reports. For more information, see *Insert HTML into Reports*.

7. Does Logi Studio include some kind of style sheet editor or do I need to get one of my own?

Logi Studio provides two ways to edit styles sheets. It includes a content-specific, internal style sheet editor, which opens in a tab inside the Studio Workspace panel, and the Class Selector tool which allows class editing and previewing within Studio. Menu

options within Studio can also launch whatever external CSS editor you may own that's associated with the .css file extension. Logi applications also *Working with Themes*, some of which are provided, and Studio includes a Theme Editor tool so you can create customized themes.

8. How can I organize my support files in the _SupportFiles folder into sub-folders?

If you create a new sub-folder, such as "images", under Support Files in Studio's Application panel, and give your support files names that begin with the sub-folder name, such as "images.logo.png", they will appear in Studio's Application panel organized into virtual sub-folders, for example, as "logo.png" beneath the "images" folder. You can also drag files into the folders and have Studio rename them automatically (see #11 below).

9. Can I use URLs or tokens to refer to external images with the Image element?

Yes. Generally, if you just supply a file name in the Image element's Caption attribute, it assumes your image file to be in the _SupportFiles folders. If you supply a complete URL ("http://somesite/myImage.gif") instead, this assumption is overridden. However, using a token in the Caption to supply a URL can be problematic, due to the timing of token evaluation. The solution is to force the Caption to be considered as a URL, by entering "http://" and then the token, like this: "http://@Data.myURLToken~". Naturally, your token must not contain the (now redundant) "http://" itself.

10. What happened to the _Images folder? I don't see it anymore in new applications.

After the release of Logi Info v10, files formerly stored in individual folders named _Images, _Stylesheets, _IncHtmls, _DataXMLs, and _Scripts were consolidated into the _SupportFiles folder, with backward compatibility provided (as long as file paths are not specified using tokens).

11. How can I organize files within the _SupportFiles folder?

Files can be organized within the _SupportFiles folder by adding sub-folders *from withinStudio's Application panel* (right-click the SupportFiles folder) and dragging files into them. This automatically adds a prefix to the file name. For example, if you created a sub-folder in Studio called "images", a file named "myLogo.png" dragged into it would be renamed "images.myLogo.png".

 Using external OS file system tools to create a sub-folder under _Support files *will not work* correctly; the sub-folder must be created within Studio. After that, you can add files to _SupportFiles externally, manually including the prefix in the name, if desired. When working with these organized files you must provide the complete file name; for example, an Image element's Caption attribute value would be "images.myLogo.png".

Other Questions

This topic provides answers to other frequently-asked and entry-level technical support questions:

1. Is it possible to schedule Logi reports to run and be distributed on a regular basis?

Yes. Logi products include the Logi Scheduler, a special Windows service or Java daemon, that manages and executes scheduled events. For more information, see *Logi Scheduler*.

2. How can I continue to manage older reports that are scheduled through the Windows Task Scheduler?

Prior to release 9.5, scheduled reports were managed using the Logi Server Manager as an interface for the Windows Task Scheduler. The Server Manager no longer includes this ability and reports are scheduled using the Logi Scheduler, but you can run the Windows Task Scheduler from the Windows Start Menu and still manage older scheduled tasks.

3. Can I store the schedules that the Logi Scheduler uses in a SQL database?

Yes. It's possible to configure the Logi Scheduler to use a MySQL, Oracle, or MS SQL Server database to store scheduled task data, rather than using the default embedded database files. This is discussed in detail in *Logi Scheduler*.

4. Can I use more than one instance of the Logi Scheduler, for example, in a clustered-server set up?

Yes. If you're storing scheduled task data in one of the supported database servers (see #3 above), you can configure multiple Logi Scheduler instances on different servers, creating a fault-tolerant arrangement.

5. Is there some way to find out at runtime what the file system path is to my application on the web server?

Yes. The @Function.AppPhysicalPath~ token will provide a fully-qualified path that includes your application folder. You can use this token as part of a file path to other folders within your application for exports, data, etc. For more information, see *Token Reference*.

6. Is there some way to find out at runtime what the URL of my application is?

Yes. One approach is to add a Session variable in your application's `Global.asax`or `rdPage.aspx`file, using the following code: `<% Session("webpath") = Request.ServerVariables("URL") %>`then reference it in your report as `@Session.webpath~`, andanother is to add this to a Label formatted as HTML: `<*script*> document.write(document.location.href); <*/script*>`(remove the asterisks which were added here to prevent the example from being interpreted as script in this document).

Glossary

A

API

API, short for Application Program Interface, is a set of routines, protocols, and tools for building software applications. In business intelligence, APIs may be used to enable end-users to directly update source systems.

Authentication

Authentication is the verification of a user's identity.

Authorization

After a user's identity has been authenticated, authorization grants or denies access to reports, columns, and records to selected users or user-groups.

B

Big Data

Refers to both the ever-growing volumes of data in use today and also to services that are specifically engineered to provide and manipulate very large data volumes.

Business Analytics

Business analytics, or business intelligence (BI), gives customers the ability to rapidly create scalable, interactive data analysis applications, and self-service capabilities users can access from anywhere and on any device.

C

Columnar Data Store

Columnar data store is a type of big data repository containing structured data in columns and rows. The main benefits are that the data can be highly compressed and is easily searchable.

CRM

A Customer Relationship Management (CRM) system is a database-based system that records a company’s daily customer-related transactions. CRMs can help customer representatives to provide better service, close more deals, and increase revenue.

CSS

Cascading Style Sheets (CSS) is a technology that allows the presentation aspects of web pages to be separated from the page content. It can be used to add "styling" (e.g. apply fonts, colors, alignment, spacing, and more) to web pages.

D

Data Discovery

Data discovery is the capability to analyze data on-the-fly and uncover insights from it.

Data Enrichment

Data enrichment is a method of preparing data to make it ready for analysis and exploitation, and can include formatting, adding calculations, joining with other data, and more.

DevNet

The Logi Developer Network website.

Drill Down

Drill Down is a capability that allows the user to get a view of the underlying or supporting data used in an analysis.

Drill Through

Drill Through is similar to Drill Down but takes it one step further by applying analysis to the underlying or supporting data.

E

Elemental Development

A development approach used in Logi Info that lets developers build feature-rich applications by using reusable, pre-built elements, rather than by writing low-level code.

F

Forecasting

A technique involving data mining and analysis leading to predictions about what will happen in the future.

G

Geo Mapping

The combination of geographic and other data to produce map visualizations, such as Google or Leaflet maps.

H

Heatmap

A Heatmap chart, sometimes called a "tree map", which uses a unique arrangement of rectangles to represent data and relationships, using color and size.

I

Interpolation

The process of evaluating a literal value match containing one or more placeholders, yielding a result in which the placeholders are replaced with their corresponding values.

J

JavaScript

JavaScript is a programming language supported by the majority of modern web browsers and used by many websites.

JDBC

Java Database Connectivity (JDBC) is an API used to access relational databases. Open Database Connectivity (ODBC) is a similar API designed for use with Java.

JSON

JavaScript Object Notation (JSON) is a lightweight data-interchange format that's easy for humans to read and write, and easy for computers to parse and generate.

K

KPI

Key Performance Indicators (KPIs) are visual indicators, in the form of color-coded shapes, which are tied to a pre-defined, critical threshold.

L

LDAP

The Lightweight Directory Access Protocol (LDAP) is an Internet protocol applications use to look up information from a server and is frequently used for containing user login information.

M

My Term

My definition

N

NoSQL

"Not only SQL" (NoSQL) is an alternative to traditional relational databases, and doesn't rely on tables and a pre-determined schema. NoSQL databases are especially useful for working with large sets of distributed data.

O

ODBC

Open Database Connectivity (ODBC) is an API used to access relational databases. Java Database Connectivity (JDBC) is a similar API designed for use with Java.

OLAP

Online Analytical Processing (OLAP) is the process of analyzing data stored in multi-dimensional "cubes".

R

REST

Representational State Transfer (REST) is a type of API used to provide interoperability between computer systems on the Internet.

S

SSM

The Self-Service Module (SSM) is a package that includes Logi Info + SSRM + Discovery or Logi Platform Services.

SSRM

The Self-Service Reporting Module (SSRM) is a Logi Info add-on module that adds special elements to Info and includes the InfoGo application.

W

Write-Back

The ability to update data sources, typically by adding, editing, or deleting data.